Interface using fast USB for characterization of SPADs

Master's Thesis

by

Jakob Essbüchl

Matriculation Number 1129145

Main Advisor

Univ.-Prof. Mag.rer.nat. Dr.techn. Horst Zimmermann

Co-Advisor

Dipl.-Ing. Dr.techn. Bernhard Goll

6th October 2024

Vienna University of Technology

Institute of Electrodynamics,

Microwave and Circuit Engineering

Abstract

Optical receivers, specifically avalanche photodiodes (APDs) and single-photon avalanche diodes (SPADs), are essential for various applications in optical communications and sensing technologies. Given the variability in their performance due to process parameters and environmental factors, efficient characterization methods are desirable. Commonly, characterization methods rely on high-speed oscilloscopes. These can be expensive and slow down the process due to data transfer limitations.

To improve efficiency, this work proposes a cost-effective USB interface for streaming digital data from such optical receivers to a PC. Utilizing the CYUSB3KIT-003 EZ-USB FX3 SuperSpeed explorer kit for USB connectivity, the interface was developed with a topology of SPAD $\Rightarrow$ comparator/level shifter $\Rightarrow$ serial-to-parallel conversion $\Rightarrow$ FX3 $\Rightarrow$ PC. Custom four-layer printed circuit boards (PCBs) were designed for each stage of the interface, including configurations for the comparator and level shifter stage, as well as two distinct serial-to-parallel conversion topologies: one based on daisy-chained shift registers and the other on cascaded demultiplexers/deserializers. Firmware for the FX3 was developed to facilitate data transfer, complemented by a simple Windows application for managing data transfers.

Performance evaluations showed that the interface could operate effectively at continuous speeds up to 3050 Mbit/s, nearing the theoretical maximum of 3200 Mbit/s of the FX3 for USB connections. However, performance varied significantly across different modules, with data integrity issues at certain speeds or conditions depending on the used topology.

A discrete component strategy was chosen over an FPGA-based approach for the serial-to-parallel conversion stage due to its perceived advantages in chip availability, ease of troubleshooting and design time; however, this decision led to significant challenges and unsatisfactory performance results. Nevertheless, it led to a more cost-effective solution, avoiding the high recurring costs associated with FPGA design software licenses.

While the developed interface is functional, its usability is hindered by data integrity concerns, necessitating further optimizations or alternative design strategies. Future work is recommended to explore FPGA-based solutions, leveraging insights gained from this thesis to enhance performance and flexibility while addressing the identified issues.

Contents

1. Introduction

Optical receivers, such as avalanche photodiodes (APDs) and single-photon avalanche diodes (SPADs) are a crucial component in modern optical communications systems, laser-based sensing technologies, quantum key distribution, biomedical imaging, and more. These devices convert incoming light into electrical signals, thereby enabling the detection and processing of optical information. However, the performance of these receivers can vary wildly with process parameters, operating conditions, and environmental factors like temperature. Therefore, it is essential to be able to efficiently and reliably characterize them.

1.1. Motivation

Characterization can be done using high-speed oscilloscopes with several GHz of bandwidth. To evaluate the data, the waveforms usually need to be transferred to a PC, which can be slow either due to limited connection speeds or the high amount of samples. Such oscilloscopes also tend to be prohibitively expensive. For many measurements, it is enough to measure the number and approximate width of pulses, the exact waveforms are not needed. As such, a digital interface that can distinguish between two voltage levels can suffice. In addition, it should be able to quickly stream data to a PC using a USB connection.

1.2. Research questions

The task of this thesis is to design a digital interface capable of streaming high-speed serial data from optical receivers to a PC using USB and evaluate its performance. To enable USB connectivity, a USB 3.2 Gen 1 development board, the *CYUSB3KIT-003 EZ-USB FX3 SuperSpeed explorer kit* [1] ('FX3' for short) shall be used. Ideally, the maximum transfer speed of 3200 Mbit/s (without protocol overhead) that the development board can reach is met.

Thus the research questions are:

1. Is it feasible to use USB to stream high-speed serial data from optical receivers with the purpose of characterizing them?

2. Are there potential challenges or limitations in implementing or using such an interface?

3. How does the system compare to using high-speed oscilloscopes for characterization?

1.3. Limitations

While a USB interface will be implemented and evaluated, it is out of the scope of this thesis to use it on actual optical receivers. In their stead, bit pattern generators will be used to simulate the incoming data stream.

Any circuit design will be done on a printed circuit board (PCB) level, i.e. using commercially available off-the-shelf parts or similar. No custom integrated circuits (IC) will be developed or manufactured.

2. State of the Art

This section gives a brief introduction to single-photon avalanche diodes (SPADs) and key metrics used to characterize their performance. Additionally, it provides a brief overview of high-speed digital signalling standards, some of which are relevant to this thesis. Finally, it mentions already existing FX3-based USB interfaces in academic literature.

2.1. Overview of SPADs

Single-photon avalanche diodes (SPADs) are highly sensitive photodetectors, capable of detecting individual photons. They are a special form of avalanche photodiode (APD), operated in Geiger mode. In this mode, the cathode-anode voltage is above breakdown, so that a single photon-generated charge carrier can build up a self-sustaining avalanche process [2, p. 87]. The avalanche leads to a current in the mA range that needs to be quenched before another photon can be detected. Quenching works by reducing the bias voltage below the breakdown voltage. Quenching circuits can be passive or active, the latter using complex circuitry to improve control as well as speed. They have a significant influence on the performance of the SPAD.

Some SPADs are gated. There, the quenching and detecting time phases are forced alternately with e.g. a clock. Such 'gated SPADs' still employ a quenching circuit but can be additionally turned 'on' or 'off' via the gater.

SPAD metrics

To evaluate the performance of a SPAD, there are several metrics that need to be measured. The following is a brief list of important metrics used to characterize SPADs:

Photon Detection Efficiency (PDE) defines the average portion of photons that successfully generate an electron-hole pair in the active area of the detector that also triggers an avalanche. It rises with excess bias voltage since the avalanche triggering probability increases in higher electric fields [3, p. 1958], [4, pp. 3895-3896].

Dark counts are avalanches triggered by thermally generated charge carriers. They are specified in the form of a *dark count rate* and increase with temperature (primary pulses) as well as excess bias voltage (secondary pulses) [2, pp. 87-88],

[3, p. 1958]. Dark count rates can vary widely, ranging from as low as 100 counts per second [3, p. 1959] to as high as the megahertz range [5]. Dark counts are measured by operating the SPAD in dark conditions. (Dark) pulses are counted and evaluated using statistical methods like the time-correlated single photon counting (TCSPC) technique [6, p. 138].

Afterpulsing is a process where charge carriers are captured in deep levels during the avalanche process. After some delay they are released and can trigger another avalanche [2, p. 88]. These pulses are correlated with the previous pulse and increase with the excess voltage. Afterpulsing can be measured with a special double-gate method, where the first gate triggers an avalanche and the second gate measures potential afterpulses [7]. Another method is generating a cross-correlation histogram between the trigger signal and detector output [8].

FWHM (full width at half maximum) is the width of a spectrum curve or distribution, measured at half the maximum value. In the context of SPADs it is commonly used to characterize the distribution timing response. FWHM for response time can be measured and evaluated using the previously mentioned TCSPC technique.

2.2. High-speed digital signalling

The need for faster and more efficient digital systems has led to the development of various high-speed signalling standards. The following is a non-comprehensive list of these standards for speeds ranging from several hundred Mbit/s to 52 Gbit/s.

HSTL: Released in 1995 as EIA/JESD8-8 [9], the 'High Speed Transceiver Logic' standard is meant to be technology independent. It allows the device supply voltage to be different from the output supply voltage. It is a single-ended standard designed for use at speeds less than 300 MHz [10, p. 1].

LVDS: 'Low Voltage Differential Signaling' is very efficient, and simple to terminate. As the name implies, LVDS is a purely differential standard. It is limited to ≈1 Gbit/s in effective performance [11, p. 1], although rates of up to 3 Gbit/s have been reported [12, p. 10].

CML: In 'current mode logic', an emitter-coupled pair of transistors are implemented in a topology that results in a voltage-controlled current switch. This switch steers the bias current to one of the two transistors. Load resistors convert the current into a voltage [13, pp. 35-37]. CML can work differentially or single-ended. CML buffers have advantages over CMOS inverters in high-speed low-voltage applications, especially in the presence of environmental noise sources like crosstalk or power rail noise [14].

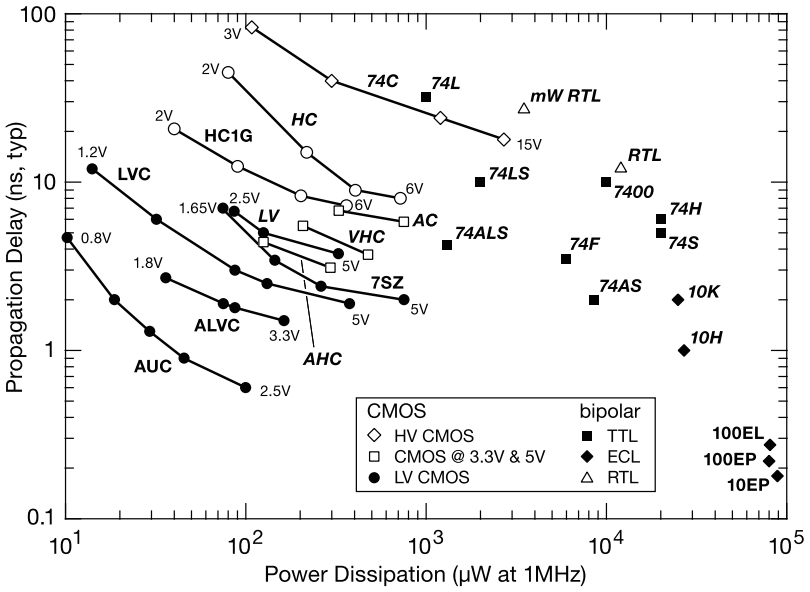


Figure 2.1.: Gate delay versus power for various logic families. Used ECL families in this thesis are 10EP and 100EP as well as LVCMOS (unknown specific logic family). Source: [17, p. 792]

ECL: With its origins in a 1956 patent [15], emitter-coupled logic (ECL) has historically been the fastest logic family. It still sees use in cases where high-speed and good noise immunity is important. Positive emitter-coupled logic (PECL) was a development that allowed to use positive supply rails instead of the negative supply rails that ECL needed. Low voltage positive emitter-coupled logic (LVPECL) was another improvement that lowered the needed supply voltage from 5 V to 3.3 V or even lower.

ECL is essentially CML with added output buffers [13, p. 48]. This changes the high output impedance of CML to a low impedance (around 5 Ω), thereby increasing the drive capability.

The biggest downsides of LVPECL are its large static power consumption compared to technologies like CMOS or TTL (fig. 2.1) and the relatively complex termination requirements. It has very low propagation delay (<500 ps), however.

JESD204: The first version of the JESD204 interface specification was released in 2006, offering a 3.125 Gbit/s maximum lane rate. A newer version, JESD204B, features maximum lane rates of 12.5 Gbit/s and support for advanced features like end-to-end deterministic latency or harmonic frame clocking [11, p. 1]. The newest version, JESD204D, released in 2023 supports lane rates of up to 58 Gbit/s with NRZ encoding and is fully backwards compatible to JESD204B [16]. In the electrical layer, the signal represents a 1.2 V CML signal [16, p. 11]. The JESD204 interface can be implemented in FPGAs but is highly complex. JESD204 is commonly used as the data interface of modern high-speed ADCs.

Standard	Voltage Swing in mV	Typical Data Rate* in Gbit/s	Power	Key Features
LVCMOS	Rail to rail	< 0.3	Medium	Widely used
HSTL	400–600	< 0.3	Medium	Good signal integrity
LVDS	350	< 1.0	Low	Low power, good noise immunity
LVPECL	100–800	< 3.2	High	Good noise immunity, high drive capability
CML	400–800	< 12.5	Medium	Low noise
JESD204D	400–800	< 58	Medium	Deterministic latency, harmonic frame clocking

*Per lane

Table 2.1.: Comparison of different high-speed signalling standards

2.3. FX3 USB interfaces

FX3-based designs in academic literature commonly follow a structure of *device* ⇒ *FPGA* ⇒ *FX3* ⇒ *PC* [18]–[20]. Devices can be image sensors, ADCs or similar. An important difference to this thesis is that the devices connected to the FPGA (and eventually the FX3) are connected via parallel buses, while in this thesis a single high-speed serial lane & clock needs to be converted to a 32-lane parallel bus to connect to the FX3.

For the serial-to-parallel conversion there exist fast ASIC-based solutions in academic literature [21]–[23]. However, these are custom integrated circuits (ICs).

3. Methodology

This section provides an insight into the developed USB interface. It details the system architecture, describing the abstract function of the circuits in block diagrams. Important parts of the circuits are described in detail on a schematic part-level. All schematics are included in their entirety in appendix D on page 71.

The focus lies on the actual solutions that were chosen and implemented. A discussion about alternatives will be had later in chapter 5, especially section 5.1.

3.1. System Overview

As outlined in section 1.1 the main goal of the system is to transfer data from a high-speed serial data source[1] to a PC using USB. The data source provides a data line and a synchronous clock line. These are then handled via the custom USB interface and adapted for USB transfer (see fig. 3.1 top part).

The custom USB interface consists of three parts (see fig. 3.1 bottom part). The approach is a partly modular one; each of the three parts is on its own PCB, allowing the use of different variants and easier revisions.

The following is a short summary of the three modules. They will be described in detail in the next section.

1. **Comparator/Level Shifter:** This module has two functions: generate a clean digital data signal and shift the voltage levels to what the next module needs. For this, a PCB was built that does both and will be called *Comparator PCB*. In many cases the comparator is redundant since the incoming signal is already conditioned, so a second variant of the PCB was made that only does the level shifting. This variant will be called *Level Shifter PCB*.

2. **Serial to Parallel Conversion:** This module converts the high-speed serial data line into 32 parallel data lines. The synchronous clock is divided by a factor of 32. The reason for this is that the FX3 chip that handles USB communication does not have a high-speed serial interface but rather a parallel interface that is up to 32 bits wide. Two different approaches were implemented: the *Shift Register PCB* and the *Demux PCB*.

[1] In practice the data source would be a SPAD but it could be any digital data source. Even analogue signals can be used in conjunction with the optional input-side comparator, with the caveat that the analogue signal will be directly converted to a digital one, losing any analogue information beyond the voltage being above or below the comparator threshold.

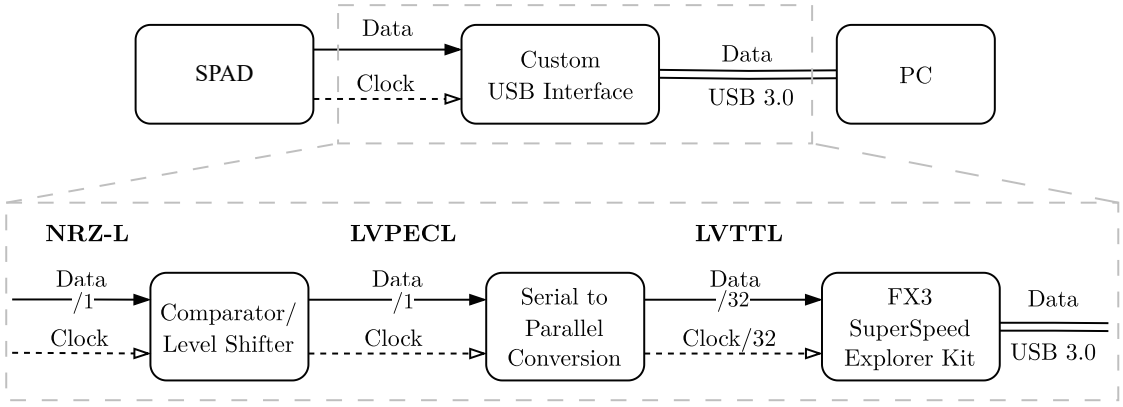


Figure 3.1.: System overview. The top block diagram shows the complete abstract structure, with the data source in form of an SPAD & gater circuit and the data sink in form of a PC. The bottom diagram further details the custom USB interface, showing its three modules. The bold text above the data lines specifies the voltage levels used.

3. **FX3 Superspeed Explorer Kit:** This module is a development kit for the Infineon EZ-USB FX3 peripheral controller. It has a lot of possible functions and applications but in the case of this thesis it serves as a data streaming device, capable of continuously transmitting data to the PC with speeds up to approximately 3 Gbit/s. It will be abbreviated as *FX3* from now on.

The next two sections go into detail on the design for each module of the custom USB interface. They will be described in backwards order (i.e. from PC to SPAD), starting with the FX3. This is because some design decisions in other modules are a result of the requirements imposed upon by the FX3.

3.2. Software Design

Code was needed for four parts:

- Most importantly, the FX3 firmware itself, handling USB communication on the hardware side.
- The Windows application, allowing to initiate and end data transfers from the PC side and write the transferred data into a file.
- One PCB included an STM32 microcontroller to configure an IC via SPI and handle user input via a button.
- Python scripts on the PC to evaluate transferred data, most importantly to calculate the bit error ratio (BER) of transferred data and gauge the performance of the interface.

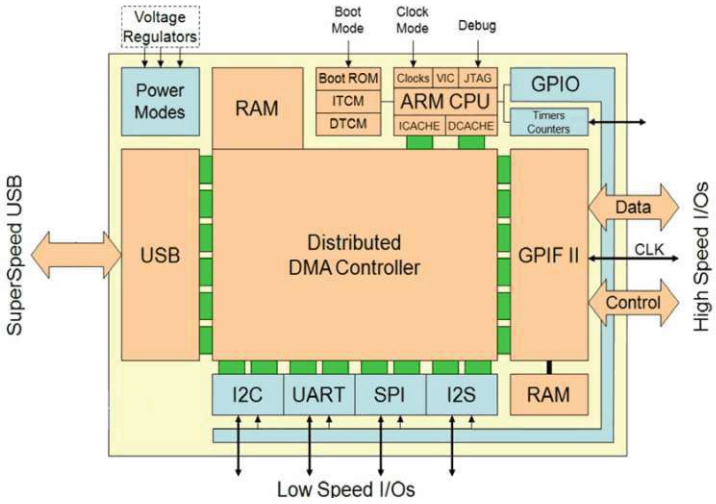


Figure 3.2.: FX3 block diagram. The green connections between peripheral blocks and the RAM (controlled by the DMA controller) are called *sockets*. Source: [24, p. 3]

These will be described in the following section. The microcontroller will be described in the respective PCB section (3.3.2).

3.2.1. FX3

Infineon's *EZ-USB FX3 peripheral controller* allows adding USB functionality to hardware. They present it as:

> FX3 has a fully configurable, parallel, general programmable interface called GPIF II, which can connect to any processor, ASIC, or FPGA. [...] It provides [...] connectivity to popular interfaces, such as asynchronous SRAM, asynchronous and synchronous address data multiplexed interfaces, and parallel ATA. FX3 has integrated the USB 3.2 Gen 1 and USB 2.0 physical layers (PHYs) along with a 32-bit ARM926EJ-S microprocessor for powerful data processing and for building custom applications. It implements an architecture that enables 375-MBps data transfer from GPIF II to the USB interface. [...] FX3 contains 512 KB or 256 KB of on-chip SRAM [...] for code and data. EZ-USB FX3 also provides interfaces to connect to serial peripherals such as UART, SPI, I2C, and I2S. [24, p. 8]

Infineon also offers the *CYUSB3KIT-003 EZ-USB FX3 SuperSpeed explorer kit* which is a development kit for the FX3 chip [1]. It allows quickly starting hardware projects with the FX3. All I/Os – most importantly the GPIF II interface – are exposed via 2.54 mm pitch pin headers on the top as well as pin sockets at the bottom (see fig. 3.3). The most important additions on the explorer kit compared to the bare FX3 chip are:

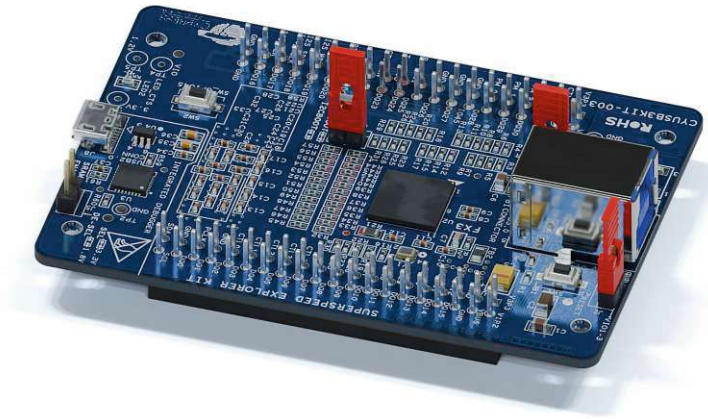


Figure 3.3.: Rendering of the FX3 explorer kit. (Main) Connection to the PC is done via the USB 3 Type-B port on the right, while the JTAG debugger is connected to the PC via the micro-USB port on the left.

- An onboard 16 Mbit SRAM[2] connected to the GPIF II interface (unused in this thesis).
- A 2 Mbit EEPROM[3] for storing the FX3 firmware between resets or power-downs. It is connected via I2C to the FX3.
- A USB-serial dual-channel bridge[4] with JTAG functionality. This acts as an onboard JTAG debugger, connected via a separate micro-USB port to the PC.

A very important fact to keep in mind is that the GPIF II interface is a *parallel* one; it allows data widths of up to 32 bit and clock speeds of up to 100 MHz, giving a maximum throughput of 3200 Mbit/s. Thus, to get the desired transfer speeds, we need to convert the high-speed serial input to a 32 bit wide parallel bus.

Since the hardware part of the FX3 module is covered wholly via the explorer kit, the rest of the FX3 section will focus on the software, namely the FX3 GPIF II state machine, the FX3 firmware and the windows application.

GPIF II state machine

General programmable interface II (GPIF II) is the main interface the FX3 to move data into and out of the FX3 [25, p. 207]. It is connected to the distributed DMA controller that connects all peripherals as well as the ARM CPU of the FX3 (see fig. 3.2). Connections between the GPIF II and the DMA controller are represented by one or several DMA sockets. The DMA fabric can directly move data between sockets (e.g. GPIF and USB) *without CPU involvement*. This allows the high throughput

[2] CY7C1062DV33
[3] M24M02-DRMN6TP
[4] CY7C65215

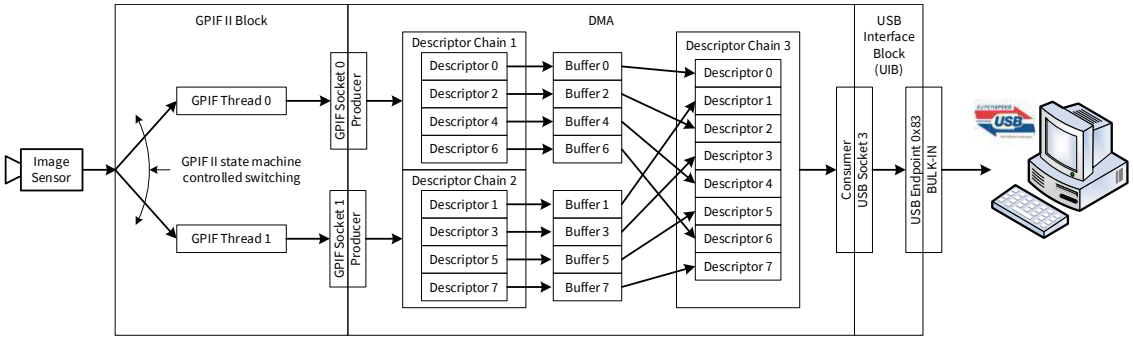


Figure 3.4.: FX3 data transfer architecture. Source: [26, p. 23]

of up to 3200 Mbit/s. Internally, the DMA controller can even achieve double that speed (6400 Mbit/s) [25, p. 32] but is limited by the speeds of the GPIF II and USB interfaces.

The GPIF II interface is controlled via the GPIF II state machine. Its configuration is loaded into 8 kB of dedicated RAM (separate from the 512 kB of embedded SRAM used by the DMA and the CPU) upon startup [24, p. 12]. The interface is very flexible, supporting various data bus widths, frequencies up to 100 MHz, and up to 16 configurable control pins (input/output or bi-directional). It can operate as a master or slave. One of the most common applications is a synchronous slave FIFO interface – a variant of which is used in this thesis. While the state machine mostly works autonomously (controlled by internal events, flags or external triggers), there is some limited capability to trigger state changes via the FX3 firmware as well.

A graphical tool called *GPIF II Designer* is used to create and configure the state machine. Each state can have several *actions* like configuring or incrementing counters, reading in from or writing data to the GPIF interface, setting or reading the state of control pins (high/low), etc. Most actions of the active state are performed on each clock cycle of the GPIF interface. State transitions occur via *events*. Examples for events are counter overflows, DMA flag changes, input GPIO state changes (as external triggers).

Before we can go into the used state machine, we need to understand some terminology in the GPIF context. For the next paragraphs please refer to fig. 3.4 for easier understanding.

- A **DMA buffer** is a section of the embedded SRAM dedicated to intermediate data storage. They can be chained within a channel to form a large continuous buffer.

- A **socket** connects peripheral blocks like GPIF II, UART, I2C or USB to the FX3 RAM, which itself is controlled by the distributed DMA controller.

- A **producer socket** writes data to a DMA buffer, a **consumer socket** reads data from a DMA buffer. Both are types of **DMA sockets**.

- A **GPIF thread** is a connection between the external data pins and a (producer) socket to a DMA descriptor chain. These threads have nothing to do with the threads of the real-time operating system (RTOS) running on the FX3 CPU.
- A **DMA descriptor** is a set of registers within the embedded SRAM and contains information about address and size of a DMA buffer. They are usually chained to point to a chain of DMA buffers.

A data transfer from GPIF to USB roughly works like this:

1. Data is sent from GPIF thread 0 to GPIF socket 0. The DMA controller writes the incoming data continuously into DMA buffer 0 connected to that GPIF socket, filling up the buffer.
2. When the buffer is full (tracked with an internal counter counting the transferred bytes) the GPIF thread switches to thread 1, thereby changing the active buffer to 1.
3. While buffer 1 is filling, the GPIF socket 0 switches to the next empty DMA buffer (buffer 2).
4. Once buffer 1 is full the GPIF thread switches back to thread 0. Now buffer 2 is being filled.
5. This alternating action goes back and forth until all DMA buffers are full. Then the whole buffer chain gets committed to the USB consumer socket. This empties the DMA buffers into the consumer socket connected to the USB endpoint.
6. Jump back to step 1 until the transfer stops

Switching DMA buffers within a socket takes some time (several milliseconds), while switching GPIF threads is done without any latency [26, p. 23], [25, p. 217]. This allows continuously streaming data on the input side without data loss, which is essential for our application. In other applications that allow flow control (e.g. transferring data from a storage medium) a single GPIF thread can also be utilized, since the data source can be briefly paused while the producer socket switches buffers.

The state machine used in this thesis is shown in fig. 3.5. It starts in an idle state called 'DMAWAIT'. When initially entering this state, two counters are initialized and configured.[5] When the user triggers the start of a data transfer on the PC, this is sent via USB to the FX3 firmware, which changes the GPIF state to 'READ_THREAD0'. Then on each clock cycle, 32 bit of data are read from the GPIF II pins and written into the DMA buffer connected to GPIF thread 0.

[5]Note that these counters are called data and address counters in the GPIF II Designer tool. They do not have that semantic function in this state machine, but are used as simple counters. The names are hardcoded into GPIF II designer.

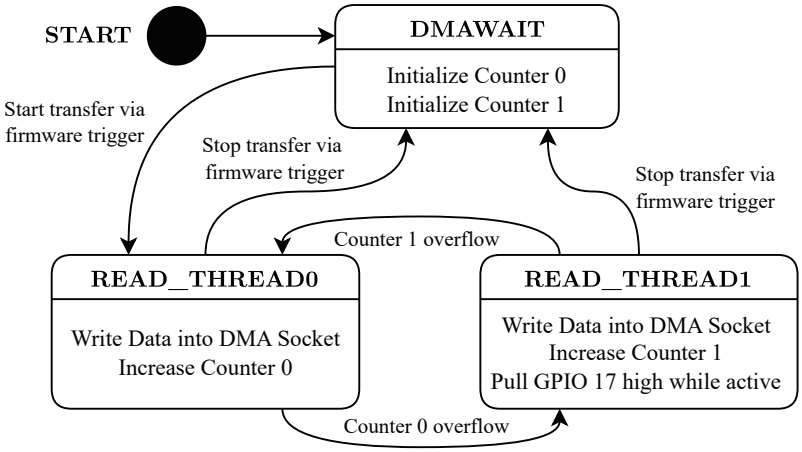


Figure 3.5.: GPIF II state machine. A screenshot of the state machine within the GPIF II Designer application is shown in the Appendix in fig. C.1 on page 67.

Once the counter overflows (indicating that the buffer is full) the state switches to 'READ_THREAD1'. Now the next DMA buffer (connected to GPIF thread 1) is filled up. For debugging reasons, a GPIO pin is also asserted (GPIO 17, pulled high) while READ_THREAD1 is active. This allows to verify that GPIF threads are actually switched as well as the speed that DMA buffers fill up (which can be calculated by checking the clock input frequency).

At any time the user can stop a running transfer on the PC, after which the FX3 firmware will switch the state back to the idle 'DMAWAIT' state.

The counters need to be initialized so that their overflow limit corresponds to a full DMA buffer. They start counting at 0 and are incremented by 1 on each (GPIF) clock cycle. The correct limit depends on the DMA buffer size as well as the GPIF data width.

$$\text{counter limit} = \frac{\text{DMA buffer size}}{\text{GPIF bus width}} - 1 \tag{3.1}$$

In this application each DMA buffer has a size of $2^{14} = 16384$ bytes while the GPIF bus width is 32 bit or 4 byte. This gives a counter limit value of 4095.

FX3 firmware

The FX3 firmware controls the ARM CPU within the FX3. Its main purpose (in this application) is to load the firmware from the onboard EEPROM on startup, initialize and set up the FX3, inlcuding its real-time operating system (RTOS). In addition, it also enumerates the device for USB, and configures as well as continuously handles DMA buffers to be used by the DMA fabric. Handling in this case means configuring and setting up DMA buffers before transfers as well as destroying the buffers upon transfer end. The actual moving of data into and out of the DMA buffers is handled

```
GpifToUsb/                          Root folder
├─ .settings/
│  └─ language.settings.xml         eclipse IDE project configuration
├─ continuous_read.cydsn/           GPIF II designer project files
│  ├─ projectfiles/
│  │  ├─ gpif2model.xml
│  │  ├─ gpif2timingsimulation.xml
│  │  └─ gpif2view.xml
│  └─ continuous_read.cyfx
├─ .cproject                        eclipse IDE project configuration
├─ .project                         eclipse IDE project configuration
├─ cyfx_gcc_startup.S               Startup code to set up stacks and interrupts
├─ cyfxbulkdscr.c                   USB enumeration descriptors
├─ cyfxgpif2config.h                Header file produced by GPIF II designer
├─ cyfxgpiftousb.c                  Main application
├─ cyfxgpiftousb.h                  Constants, endpoint and socket definitions
├─ cyfxtx.c                         Exception handlers & memory allocation routines
├─ makefile
├─ makefile.init
└─ readme.txt
```

Figure 3.6.: File and directory structure of the FX3 firmware. The content of the files is briefly described in the grey text adjacent to the filenames.

by the DMA fabric and the GPIF II state machine. It also handles communication with the PC application via USB, initiating and stopping data transfers on user request.

The code used in this thesis can be accessed online on github via `https://github.com/Fivefold/fx3_usb_interface`. Additionally, it will be hosted on the Internet Archive via `https://archive.org/details/fx3_usb_interface` and is also included on a CD in the physical versions of this work.

The basis for the firmware code is the `cyfxbulksrcsink` example found in the FX3 software development kit (SDK) at the path `/1.3/firmware/basic_examples`. This example was used and modified to create the current implementation.

An overview of the firmware files can be seen in fig. 3.6. The GPIF II designer files are in a separate folder `continuous_read.cydsn`. These contain the project files used to configure and compile the header file of the GPIF II state machine described in the previous section.

```
57 /* Burst length in 1 KB packets. Only applicable to USB 3.0. */
58 #ifndef CY_FX_EP_BURST_LENGTH
59 #define CY_FX_EP_BURST_LENGTH           (16)
60 #endif
61
62 /* Size of DMA buffers (in bytes) used by the application. */
63 #ifndef CY_FX_DMA_BUF_SIZE
64 #define CY_FX_DMA_BUF_SIZE              (16384)
65 #endif
66
67 /* Number of DMA buffers to be used on the channel. */
68 #ifndef CY_FX_DMA_BUF_COUNT
69 #define CY_FX_DMA_BUF_COUNT             (4)
70 #endif
```

Listing 1: Important configuration options in `cyfxgpiftousb.h`

`cyfx_gcc_startup.S`, `cyfxtx.c` as well as the `makefile` can be directly used without modification from the SDK example. They roughly fit the role of a software library. For this reason they won't be explained in detail here.
`cyfxbulkdscr.c` has slightly modified USB enumeration descriptors but stays largely the same as the SDK example. The `cyfxgpiftousb` header and source files are the most important files since they contain the bulk of the application. They will be explained in more detail, with a special focus on *changes* that were done to the SDK example.

`cyfxgpiftousb.h` contains various constant definitions and configuration data, among them which consumer and producer sockets to use and how many DMA buffers at which size to use.
DMA buffers take up space within the RAM of the FX3, thus there is a limit on total buffer size. In the case of the FX3 explorer kit (which has 512 kB RAM) the maximum space available for DMA buffers is 224 kB [27].

For this application a DMA buffer size of 16 kB and a buffer count of 4 was determined to be ideal (see listing 1). Note that we have *two* GPIF threads, so the total amount of DMA buffer space is $16\,\text{kB} \cdot 4 \cdot 2 = 128\,\text{kB}$.

Achieving maximum throughput is not necessarily gained by using all the available space. Instead the parameters transfer burst length, DMA buffer size and amount can be varied to affect the throughput [28, p. 9]. The achievable throughput is not only determined by these settings but also by the host system (PC). Usually several USB ports share a single USB host with shared bandwidth. Further down the signal path several devices and interfaces share bandwidth in connection to the CPU, usually via a shared PCIe interface. Thus, available bandwidth on the host system is an important limiting factor [28, p. 8].

`cyfxgpiftousb.c` is the main source file, containing the application code. This

is where most changes (compared to the SDK example) were done and is the most important part of the firmware. As such, it will be explained in detail, starting with explaining each function in the order they appear in the code. • is used for unchanged functions or trivial changes like renamed variables, ⊙ for functions with minor changes, ⊛ for functions with major changes, and ⊕ for new functions. Changes will be explained in *italic font*.

⊙ CyFxAppErrorHandler: Error handler function. *Resets the FX3 device when an error is raised.*

• CyFxApplnDebugInit: Initializes the debug module. All debug prints are routed to the UART and can be seen using a UART console[6] running at 115200 baud rate.

⊕ GpifToUsbDmaCallback: DMA callback function that is run when DMA event happens. Increases the DMA event counters. DMA events happen when a DMA buffer is filled (producer event) or is read out (consumer event). Events are also sent to the DMA socket (producer socket ⇔ consumer socket). The signalling is handled by the DMA fabric without CPU involvement [29, p. 69].

• LoopBackDmaCallback: Handles loop-back transfers from the firmware side. When a DMA buffer gets full the data gets copied over and sent to the consumer endpoint. After this, the DMA buffer gets discarded, i.e. emptied. The loop-back function is from the SDK example but is unused in our application.

⊛ CyFxApplnEpCallback: USB endpoint specific event callback. *This keeps track of the number of USB endpoint events by counting them.*

⊛ CyFxApplnStart: Starts the application. This is called when a SET_CONF event is received from the USB host. As this is a long function the function flow will be described in more detail:

1. Endpoints are configured. Their type is set to CY_U3P_EP_BULK (for bulk transfers), the endpoint packet size and USB burst length are configured. *Disables automatic switch to USB low power modes U1 and U2 to optimize throughput.*
2. The endpoint memory is flushed.
3. DMA channels are set up. *For this application a multi-DMA AUTO channel is used, Multi-DMA meaning more than one producer socket. These are mapped to our two GPIF threads on the input side, and to the USB consumer socket on the output side. This allows gapless transfers, as explained in section 3.2.1. An AUTO channel lets the DMA transfers be handled by the DMA fabric and the GPIF II state machine.*

[6] The UART console is accessible via the Micro USB port that the JTAG debugger also uses (see fig. 3.3).

4. DMA channels for the loop-back function are set up. The loop-back function is from the SDK example and unused in our application.
5. The GPIF state machine is loaded.
6. A flag is updated to notify the RTOS application thread that initialization is complete.

⊙ CyFxApplnStop: Stops the application. Whenever a RESET or DISCONNECT event is received from the USB host, this function is called. Endpoints are disabled, their memory is flushed, and the *multi*-DMA channels are destroyed.

⊛ CyFxApplnUSBSetupCB: Callback function to handle USB *requests*. Examples are *getting firmware version and revision number, USB event log data and – most importantly – starting and stopping data transfers via the USB host. This callback function triggers the transfers of the GPIF II state machine.*

⊙ CyFxApplnUSBEventCB: Callback function to handle USB *events*. Examples for USB events are connection, reset or disconnection of the USB port. *Prints additional debug data.*

• CyFxApplnLPMRqtCB: Callback function to handle USB Link Power Management (LPM) requests from USB 3.0 hosts. The API invokes this function whenever a state change from U0 → U1 or U0 → U2 happens. U0, U1, and U2 represent power states, where U0 is the active state and U1 and U2 are power saving states. U2 conserves more power at the cost of higher exit latency [30, pp. 188-192].

⊙ CyFxApplnInit: Initializes the USB module, setting the enumeration descriptors. *Also calls the* `CyU3PPibInit` *function, which powers up the P-Port interface block. This block is used by the GPIF II interface.*

⊕ printGpifToUSBDMAStats: Helper function that prints the producer and consumer events to the debug console and resets the event counters to 0.

⊛ CyFxAppThread_Entry: Entry function for the RTOS thread. Calls the debug console and application initialization functions. While the application is active, it repeatedly calls printGpifToUSBDMAStats to print debug information and tries to keep the device in U0 power state. Resets the device upon host request. *Removed a bunch of redundant USB setup commands.*

• CyFxApplicationDefine: Defines the application thread that is executed by the RTOS

⊙ main: initializes the FX3, sets up the caches, configures the I/Os and starts the RTOS kernel. *No firmware GPIOs are enabled.*[7]

[7] One GPIO is used and controlled by the GPIF II interface, as explained in section 3.2.1.

The easiest way to compile the code is to use the provided toolchain based on the popular eclipse IDE. Infineon calls their adaption EZ USB Suite. It is part of the FX3 SDK.[8]

Windows application

Data transfers are controlled by the user from the PC side. For this a small Windows application called *CollectData.exe* is used. It is part of the code examples of the *SuperSpeed Device Design By Example* book [25], officially endorsed by Infineon.

The code examples from the book (not to be confused with the FX3 SDK code examples) are available online under `https://www.cypress.com/fx3book`. They include applications like *CollectData.exe* (see fig. 3.7), both as a precompiled *.exe* as well as the Visual Studio projects used to compile them. No modification of the code was needed; the custom FX3 firmware was written to work directly with *CollectData.exe*.

The application is easy to use; it automatically detects any connected FX3 devices with the correct firmware[9]. Then data can either be streamed and discarded (to test the achievable transfer rate, shown at the bottom of the application window) or a file can be chosen/created via a dialogue for the data to be saved into. A transfer is started with a click on the 'Start Data Transfer' button and either stopped manually by clicking on the same button or by letting the file transfer timeout be reached.

All transferred data that is written into a file by *CollectData.exe* is stored in binary format, in little-endian order.

3.3. Hardware Design

The following pages describe the design and function of the self-designed hardware, i.e. the serial-to-parallel conversion and comparator/level shifter modules of the interface (as previously seen in fig. 3.1).

3.3.1. General hardware design considerations

Some hardware design considerations are common to all PCBs of this thesis and thus are bundled here.

LVPECL termination voltage

The Shift Register and Demux PCBs both make heavy use of LVPECL logic levels. The Demux PCB has 30 LVPECL terminations (counting differential pairs as two

[8] For compiling instructions, see the included code on the CD (in the physical versions) or visit github via `https://github.com/Fivefold/fx3_usb_interface` or the Internet Archive via `https://archive.org/details/fx3_usb_interface`.

[9] Detection works by looking for a hardcoded vendor ID (VID) and product ID (PID) that corresponds to a 'Cypress USB StreamerExample' device.

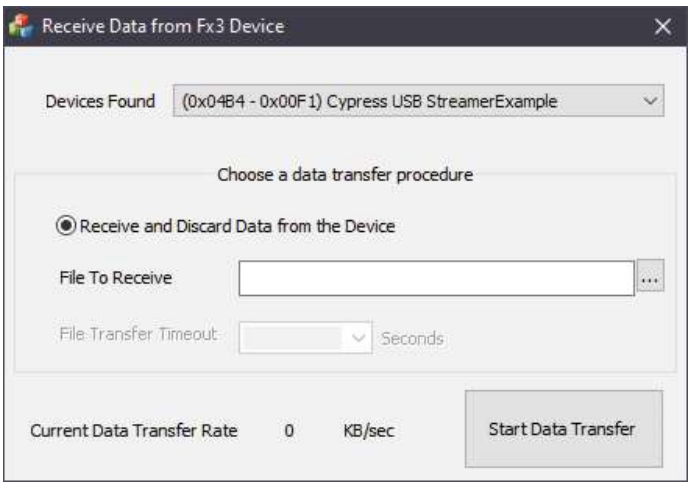


Figure 3.7.: Windows application *CollectData.exe*. The application allows to initiate and stop data transfers from the FX3 to the PC. The user can choose to write the transferred data into a file. Writing into a file also allows to optionally define timeout in seconds.

terminations since they are driven by two ECL drivers). The Shift Register PCB has 140 LVPECL terminations. Each LVPECL termination sinks about 14 mA on average, with about 22 mA when driving a logic HIGH [31, p. 9]. This gives the Shift Register PCB a worst-case termination current of about 3 A or several watts of power, necessitating active cooling. The Demux PCB with its 30 terminations only gets about 660 mA termination current, with passive cooling being sufficient.

The chosen termination method for PECL uses the standard termination. This termination method requires the supply of a termination voltage $V_T = V_{CC} - 2\,\mathrm{V} = 1.3\,\mathrm{V}$. This voltage is supplied on both boards with a modified low dropout regulator (LDO) regulator design [32]. It is paramount that the regulator can *sink* current. Thus, the design uses a *negative* regulator due to its inherent current sinking capability[10]. Usually, a negative voltage regulator outputs a negative voltage. By setting the reference potential of the regulator to V_{CC}, a positive output voltage can be achieved. This is done by connecting the GND pin of the LDO to V_{CC}.

The used LDO[11] can sink up to 400mA of current, which is not enough in this application. However, a simple modification can greatly increase the current capability. The key is to use an external PNP power transistor, its base connected to the output pin of the LDO (fig. 3.8). This way, the IC sinks the transistor base current, but maintains regulation though its feedback loop. The PNP transistor however sinks the majority of the current through its collector. This allows sinking up to several ampere, depending on the chosen transistor and cooling. No specific transistor is needed as long as it is PNP and is high-gain. Darlington transistors should be avoided, however, as their V_{BE} can be higher than the 1.3 V output voltage. To set the operating point

[10]Do not use a positive regulator unless its one of the few types that can sink current.
[11]MAX1735

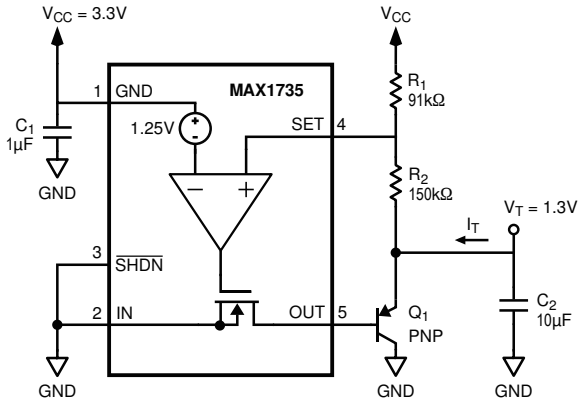


Figure 3.8.: Modified LDO for LVPECL termination voltage.

and thus the output voltage, the correct feedback voltage needs to be applied to the SET pin of the LDO. This is done using a standard voltage divider, the values of which are calculated in eq. (3.2)–(3.6).

$$V_{R_1} = 1.25\,\mathrm{V} \tag{3.2}$$

$$V_T = V_{CC} - 2\,\mathrm{V} \tag{3.3}$$

$$V_{CC} - V_T = V_{R_1} \cdot \frac{R_1 + R_2}{R1} \tag{3.4}$$

$$2\,\mathrm{V} = 1.25\,\mathrm{V} \cdot \frac{R_1 + R_2}{R1} \tag{3.5}$$

$$0.6 \cdot R_1 = R_2 \tag{3.6}$$

PCB Stackup

All PCBs use a four-layer stack configuration (see fig. 3.9). Signals are routed on the top and bottom layers. These layers are also used to route the LVPECL termination voltage if space allows. In especially space constrained areas the lower inner layer (called In2.Cu in fig. 3.9) is used to route the termination voltage as well. The upper inner Layer (In1.Cu) contains a ground plane. The lower inner layer (In2.Cu) is primarily used for a power plane but also contains large ground islands in areas where microstrips had to be routed on the bottom layer (see fig. 3.10). These islands are stitched to the ground plane with many vias to keep tight coupling and low inductance.

While it is possible to use a power plane as the reference plane for a microstrip, it requires decoupling capacitors to allow the return current to switch planes. A lower inductance alternative is to keep both reference planes at the same voltage, because then vias can be used, which are much lower inductance than the route via ⇒ decoupling capacitor ⇒ via [33, pp. 631-632]. Not using stitching capacitors or,

Function	Layers	Thickness mm	Thickness mil
	F.Mask	0.010	0.39
Signal	F.Cu	0.035	1.38
		0.099	3.91
Ground plane	In1.Cu	0.015	0.60
		1.265	49.80
Power plane & ground islands	In2.Cu	0.015	0.60
		0.099	3.91
Signal	B.Cu	0.035	1.38
	B.Mask	0.010	0.39

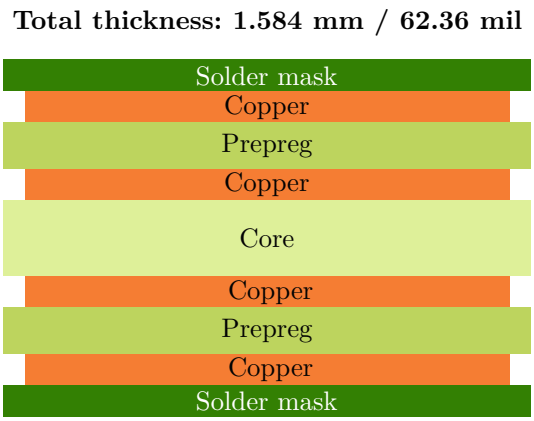


Figure 3.9.: Four-layer PCB stackup used in all designs.

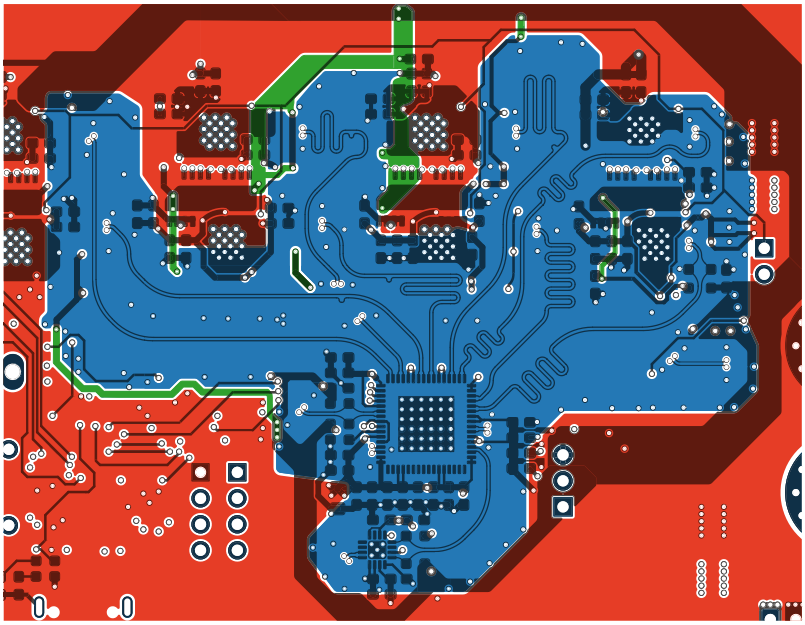


Figure 3.10.: Example of a ground island on the Shift Register PCB. The displayed layer is the inner lower layer (In2.Cu). ■ Red areas are the 3.3 V supply plane. ■ Blue areas are ground. ■ Green areas are the 1.3 V LVPECL termination voltage. Copper on the bottom layer is displayed in a semi-transparent dark overlay, showing signals as well as the routed termination voltage.

if possible, vias can significantly increase inductance and thus radiated emissions. Tests by Smith show a difference of up to 30 dB [33, p. 632], [34].

Routing

All traces that carry signals of higher frequencies ($> 50\,\text{MHz}$ fundamental frequency) were designed as microstrips and impedance matched for a single-ended impedance of $50\,\Omega$ or a differential impedance of $100\,\Omega$. Any higher-frequency signals that are supposed to arrive at the same time, e.g. data signals of parallel buses or distributed clocks were length-matched if needed.

3.3.2. Shift Register PCB

As noted in section 3.2.1, the FX3 needs a 32 bit parallel interface. The output of the SPAD gater circuit as well as the Comparator/Level Shifter PCBs is a single high-speed differential serial signal, however. A serial-to-parallel conversion is needed. The Shift Register PCB is one implemented solution to tackle this. The full schematics are available in appendix D.3 on page 76.

A block diagram of the circuit can be seen in fig. 3.11. The data enters a special high-speed 8 bit shift register[12] to get split into 8 parallel data lines. Because 32 lines are needed, three additional shift registers are daisy-chained to the first.

The internal structure of the shift register does not feature an output latch. This results in each individual data line switching to the next bit at the initial high frequency. We now have 32 copies of the initial data stream, shifted by one bit relatively to each other. Considering signal integrity but also ICs further down the line the redundant edges in the shifted outputs should be removed, thereby matching the speed of the output signal required by the FX3's parallel input.

Fortunately, the shift register also has a parallel-in parallel-out operating mode. When a (rising) clock edge comes in, all inputs of the shift register at the time of the edge get loaded and latched to the output until the next clock edge. Combined with a clock divided by 32 this allows the shift registers to act as a latch that can handle the high input speeds. This removes the redundant edges. Thus, a second batch of four shift registers were cascaded behind the first ones. This gives four shift registers acting as *shifters*, requiring the incoming high-speed clock to be fanned out to four chips and four shift registers acting as *latches*, requiring a 1/32 clock fanned out to four chips.

The clock distribution network hinges on one central part: a programmable clock divider IC[13] that also functions as a clock fanout as well as a configurable clock delay. This chip will henceforth be referred to as *clock IC*.

Thanks to the versatility of this chip, most of the clock-related requirements can be met with just one IC. This comes at the cost of additional complexity; the clock IC

[12] MC100EP142MNG
[13] 8V79S680NLGI

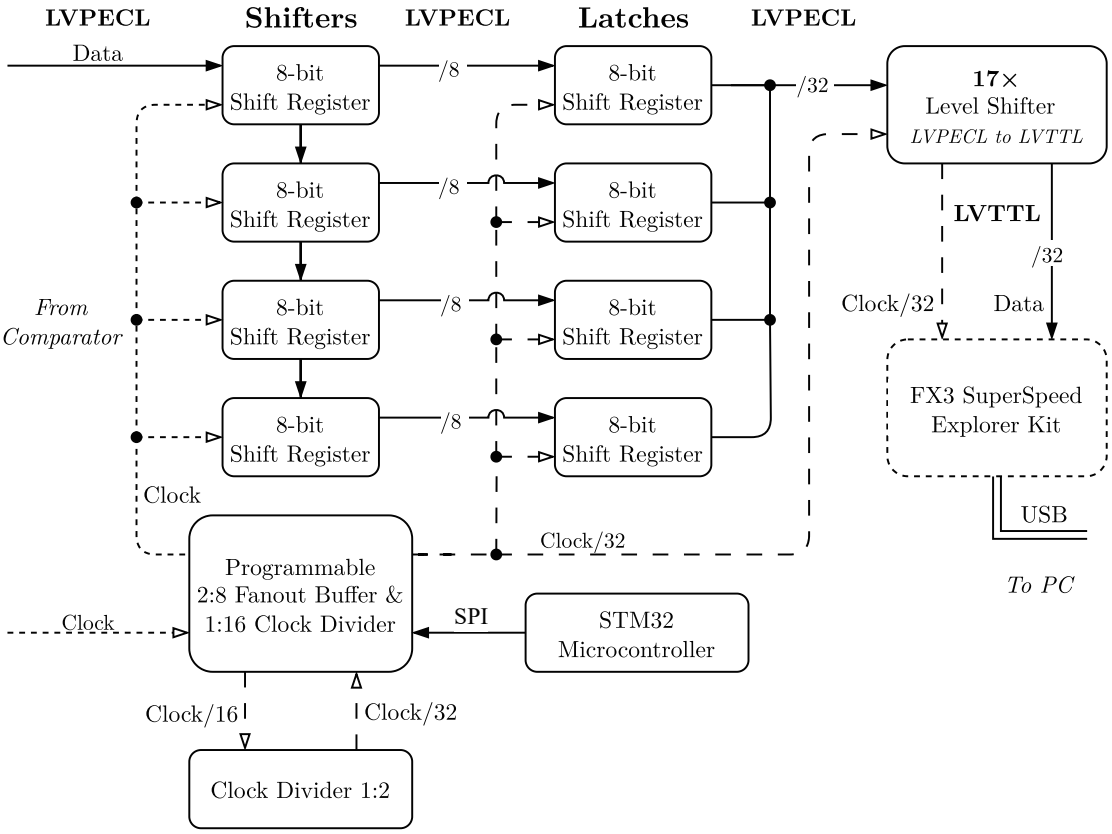


Figure 3.11.: Block diagram of the Shift Register PCB's circuit. Bold uppercase text specifies the used voltage line levels. Text like '/8' on a line represents a bus of e.g. 8 parallel lines (counting differential signals as 1). Note that the 'FX3 Superspeed Explorer Kit' is stacked on top of the PCB, connected through stacking connectors. It is, however, not considered part of the Shift Register PCB.

needs to be configured via SPI after power-on. It also requires a calibration procedure to adjust the internal delay circuitry [35, pp. 8, 13]. Configuration and calibration is done through an STM32 microcontroller. To calibrate, a clock must be applied to the differential clock inputs. Then, a button (marked with CAL on the PCB) needs to be pushed to start the calibration procedure. Calibration takes less than a second, after which the clock outputs are enabled. This is indicated to the user by an onboard LED (marked with 'READY' on the PCB silkscreen) being constantly lit. Before calibration it is slowly blinking.

The clock IC has two differential clock inputs; a high-speed one allowing up to 3000 MHz and a low-speed one with up to 100 MHz. Each input has eight differential outputs respectively and each output can be configured with a delay as well as a division factor of up to 16. To achieve the needed division factor of 32, one of the high-speed outputs is fed into a 1:2 clock divider[14] and then back to the low-speed input of the clock IC. Four of the high-speed outputs are routed to the shifters and four of the low-speed outputs each are routed to the latches. Six differential output pairs are left open, as this does not affect performance but reduces power consumption [36, p. 2].

Since the STM32 microcontroller had unused UART and I2C interfaces as well as I/Os, these were routed to optional pin headers. A USB-C port allows flashing firmware without a dedicated debugger or programmer through the STM32CubeProgrammer application. It does not allow debugging however, so for development STM32's Serial Wire Debug (SWD) interface was routed to a pin header as well.
While it would have been possible to cover all the functions of the microcontroller on the FX3, the decision was made to go with a dedicated microcontroller. This allows for separate testing of the PCB, independently of the FX3. Combining them would have made testing and debugging significantly more complicated.

The STM32 microcontroller's code is done in C, using STMicroelectronics' HAL API. This allowed quickly getting the SPI interface up and running. The SPI interface of the clock IC is a 3-wire half-duplex variant, i.e. there is only one wire for data transmission in both directions. It also uses 1.8V, requiring a level translator to the STM32's 3.3V SPI interface.

The PCB has a size of 128 by 100 mm. It can be seen in figs. 3.12 and 3.13. Power can be supplied to the board either via 4 mm banana sockets or through screw terminals. It also features a 2-pin JST-XH socket to supply power to the much smaller Comparator or Level Shifter PCBs. There are also a 2-pin JST-XH as well as a standard 4-pin molex KK 254 connector to connect a 12 V fan for active cooling. The 12 V is generated through a small boost converter circuit on the bottom side of the PCB.

[14] SY89874UMG

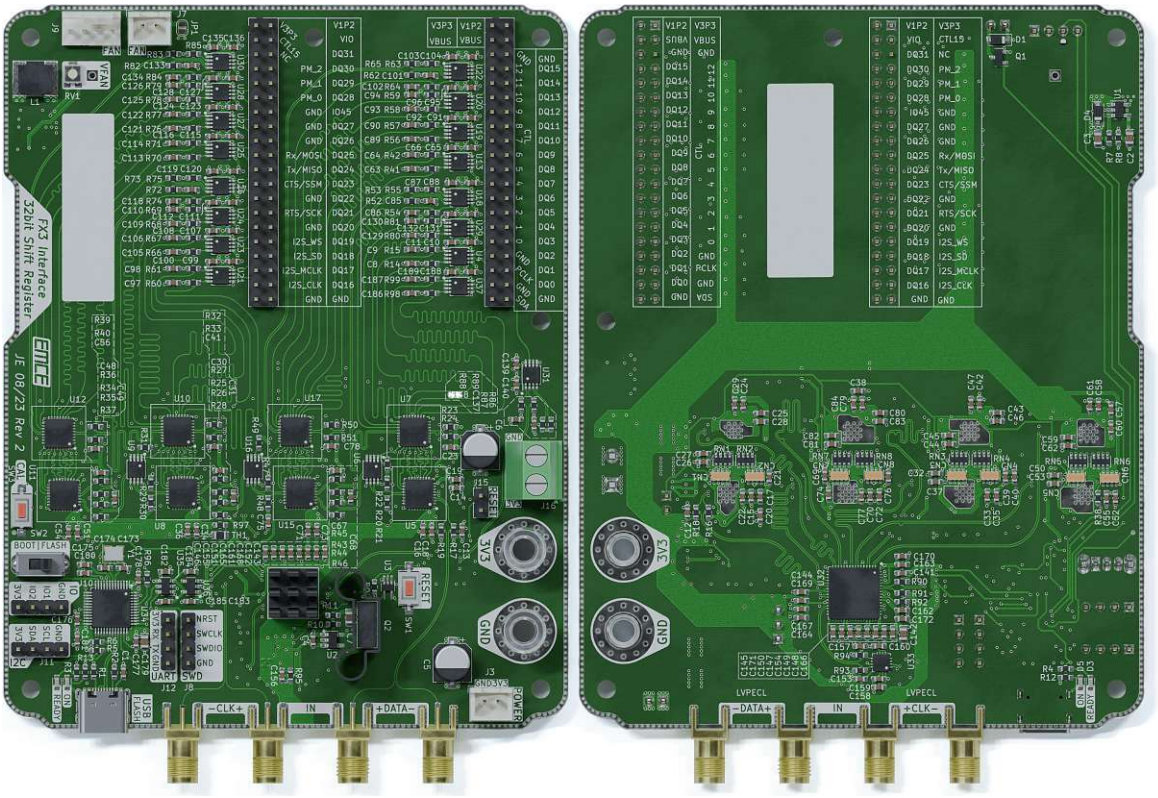


Figure 3.12.: Top and bottom renderings of the Shift Register PCB (not to scale). The FX3 Explorer Kit (not shown) is stacked on top of the pin headers. The horizontal top row of small square ICs are the latches, the bottom row are the shifters. The big square IC on the bottom side is the programmable clock divider. The square IC on the bottom left of the image is the STM32 microcontroller.

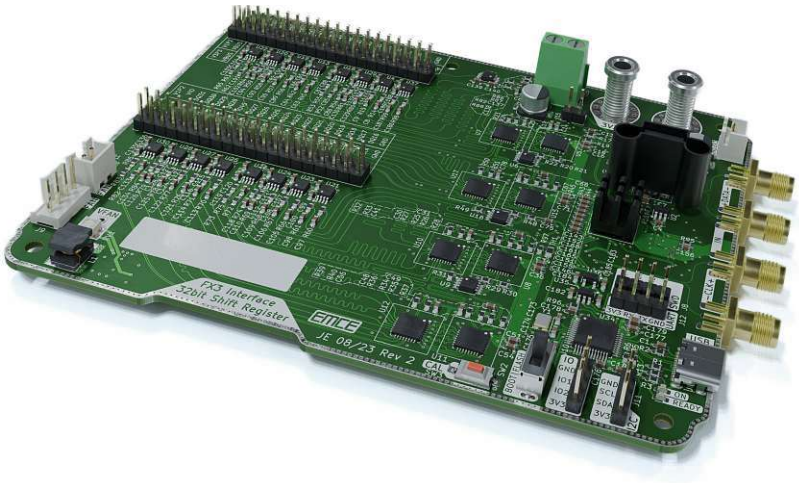


Figure 3.13.: Perspective rendering of the Shift Register PCB.

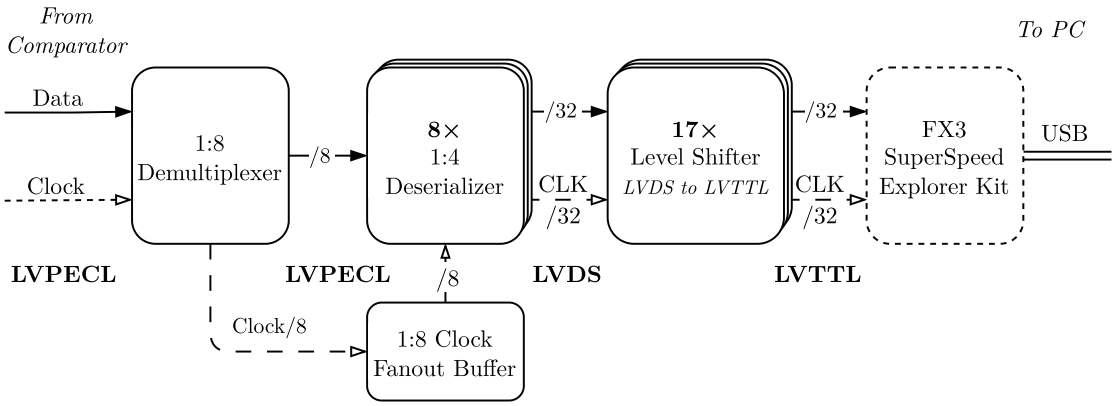


Figure 3.14.: Block diagram of the Demux PCB. Bold uppercase text specifies the used voltage line levels. Text like '/8' on a line represents a bus of e.g. 8 parallel lines (counting differential signals as 1). Note that the 'FX3 Superspeed Explorer Kit' is stacked on top of the PCB, connected through stacking connectors. It is, however, not considered part of the Demux PCB.

3.3.3. Demux PCB

A different approach to the serial-to-parallel conversion of the Shift Register PCB is the Demux PCB. This was created as an alternative, after the Shift Register PCB showed some performance issues (more on this in later sections). Where the Shift Register PCB used shift registers as the core component type to handle the conversion, the Demux PCB uses a 1:8 demultiplexer[15] and eight 1:4 deserializers[16], one for each output of the demultiplexer (see fig. 3.14). A big advantage is that these chips also divide and output the clock, so there is no need for additional clock division. The only clock-related IC is a clock fanout to share the 1:8 divided output clock of the first demultiplexer with the eight 1:4 deserializers. This simplifies the circuit significantly compared to the Shift Register PCB.

The initial 1:8 demultiplexer requires the reset signal to be asserted for some time upon power-up because its internal flip-flops start in random states [37, p. 8]. To avoid having the user press a reset button on each startup, a supervisor IC[17] was added. This chip tracks the power supply voltage and keeps the reset signal asserted for some time after V_{CC} rises above a threshold of 2.93 V. A reset button was still added for debugging purposes, however.

The PCB has a size of 100 by 100 mm, a reduction in size of $\approx 28\,\%$ compared to the Shift Register PCB. It can be seen in figs. 3.16 and 3.17. Just like the Shift Register PCB, it shares the 4 mm banana socket and screw terminal power connections as well as the 2-pin JST-XH socket to share power with the Comparator or Shift Register

[15] MC100EP445MNG
[16] MAX3681
[17] AP1702

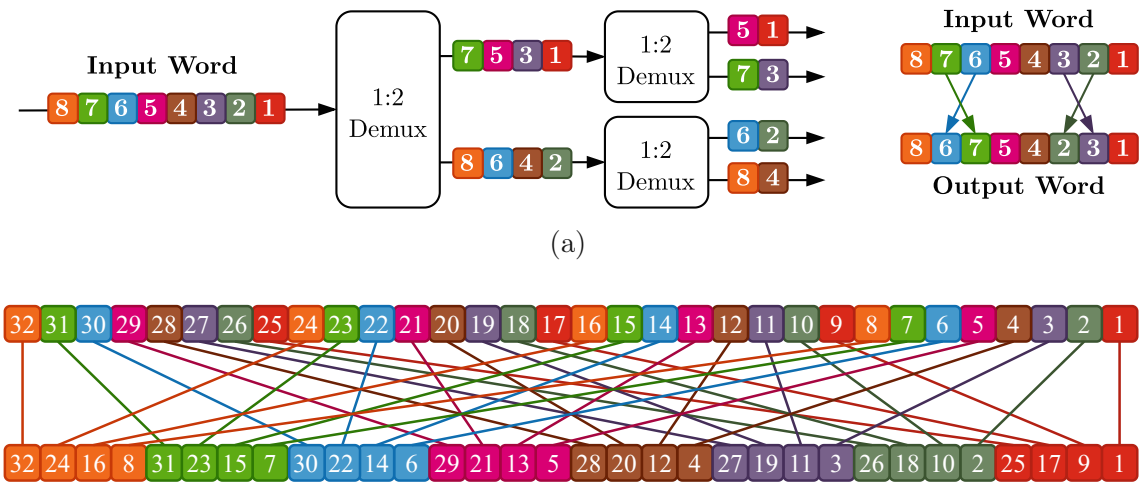


Figure 3.15.: The cascaded demultiplexer problem. (a) Simplified example of cascaded demultiplexers. Coloured boxes represent individual bits, the numbers denote the (input word) bit position. Note how the bit positions switch in the output word due to the cascading of demultiplexers. (b) Actual situation on the Demux PCB with an initial 1:8 division and 8 cascaded 1:4 divisions.

PCBs. There's also a 2-pin JST-XH socket with its dedicated 12 V boost converted for a cooling fan; however this was included just in case and the PCB ended up not needing active cooling.

The cascaded structure of demultiplexer and deserializers creates an initially unanticipated issue I call the 'cascaded demultiplexer problem', shown in fig. 3.15. As can be seen in fig. 3.15a, cascading demultiplexers leads to bits changing positions in the output compared to the input.[18] In the case of the Demux PCB, these changes are substantial (see fig. 3.15b). It is a complicated-looking pattern, however it *is* a pattern and can be corrected in software algorithmically. In theory, it would also be possible to correct this in hardware by adapting the routing between the output pins and the level shifter ICs (the last ICs in the signal chain of the PCB). The large number of crossing traces would make routing very challenging though, even more so for microstrips, so it was decided to correct for this in software.

[18]Same thing applies to cascaded deserializers or a combination of demultiplexers and deserializers.

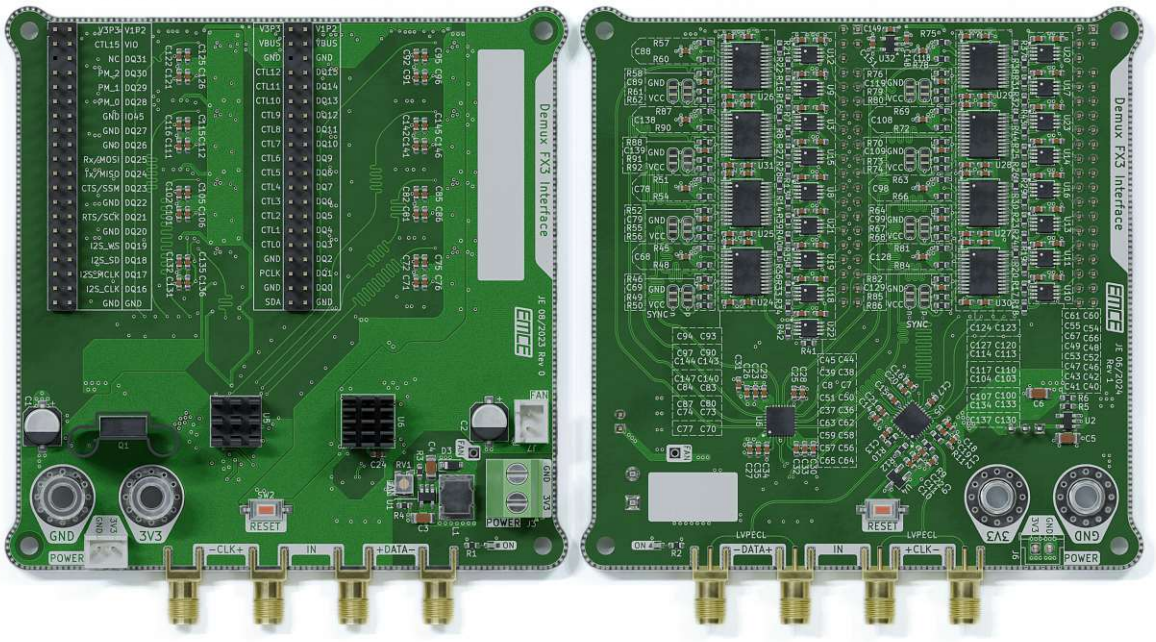


Figure 3.16.: Top and bottom renderings of the Demux PCB (not to scale). The FX3 Explorer Kit (not shown) is stacked on top of the pin headers. Next to the pin headers are the level shifters. The 8 large rectangular chips are the 1:4 deserializers. The two square ICs are the clock fanout (left) and the 1:8 demultiplexer (right). The big round connectors in the centre of the image are 4 mm banana sockets.

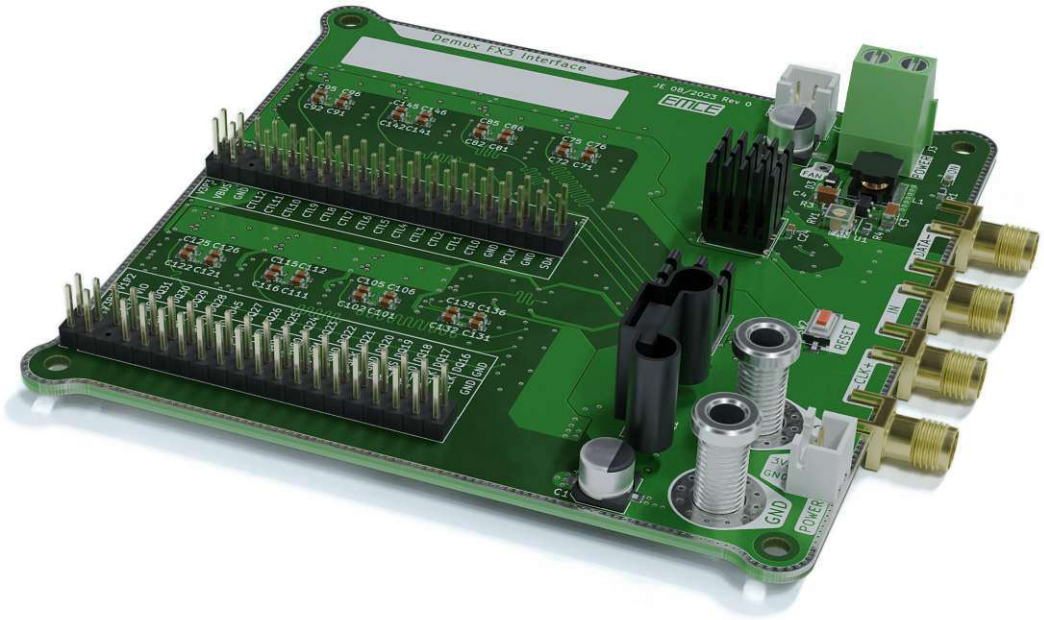


Figure 3.17.: Perspective rendering of the Demux PCB.

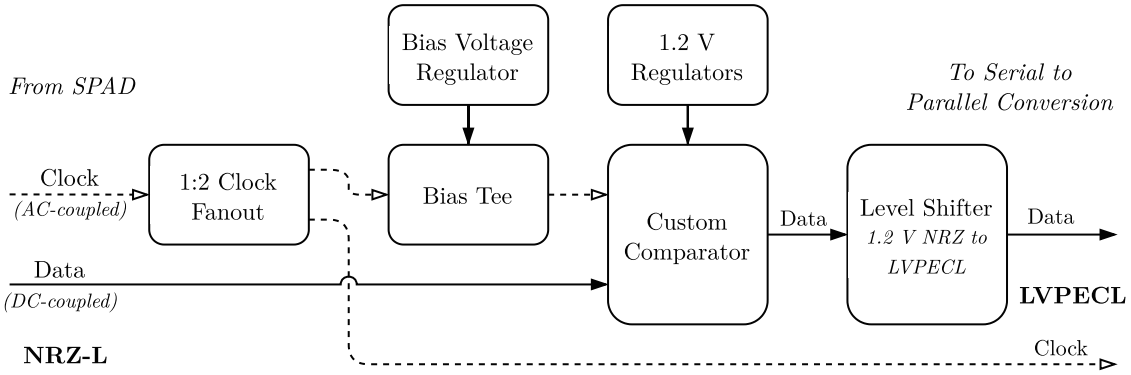


Figure 3.18.: Block diagram of the Comparator PCB. Bold uppercase text specifies the used voltage line levels.

3.3.4. Comparator PCB

The previously described Shift Register and Demux PCBs require LVPECL voltage levels on their inputs. The SPADs that are to be characterized using the USB interface use different voltage levels, however. For this reason another PCB module was created with three objectives:

1. Create clean, sharp state transitions between logic states.
2. Shift voltage levels to LVPECL levels.
3. Deliver enough voltage swing for LVPECL receivers, especially for SPAD or APD receivers, which only deliver small pulse amplitude levels down to around 100 mV.

Objectives #1 and #3 are handled through a custom in-house 7 GHz comparator, manufactured in 65 nm CMOS technology [38, pp. 213-225]. The comparator does not output LVPECL voltage levels either, however. Instead, it outputs differential data with voltage levels of 1.2 V/0 V for logic high and low, respectively. This is not a typical voltage level standard for high-speed serial communication. Thus, objective #2 is handled with a special 'Anything-to-LVPECL Translator'[19] that can handle both the uncommon voltage levels as well as the speed (up to 3200 Mbit/s). While it is possible to use bias tees for level shifting, this approach is only suitable for clock signals in this application. The reason is that the data signals can contain long sequences of continuous 0s or 1s, which are encoded using non-return-to-zero level (NRZL). In NRZL, a logic-level high represents a binary 1, and a logic-level low represents a binary 0.

The problem with using bias tees for data is the inherent DC signal component in this encoding scheme. Without DC-coupling, the bias tee output could drift to the bias voltage level, leading to errors in the transmitted bits.

[19] MAX9376

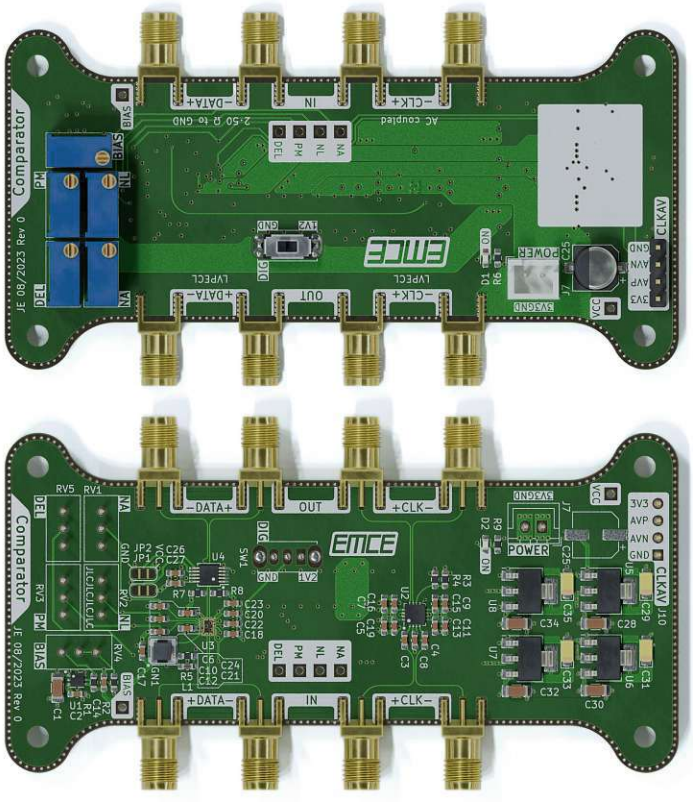


Figure 3.19.: Top and bottom renderings of the Comparator PCB (not to scale). The custom comparator IC is on the left (U3). The level shifter IC is directly above it (U4).

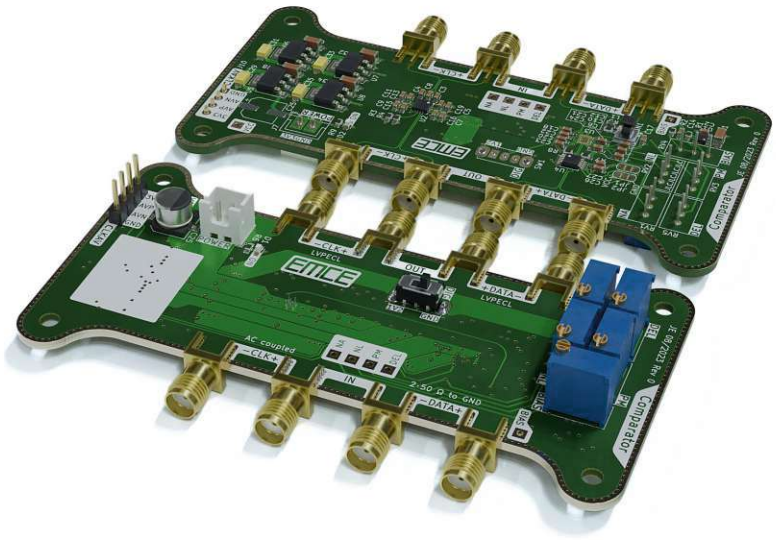


Figure 3.20.: Perspective rendering of the Comparator PCB.

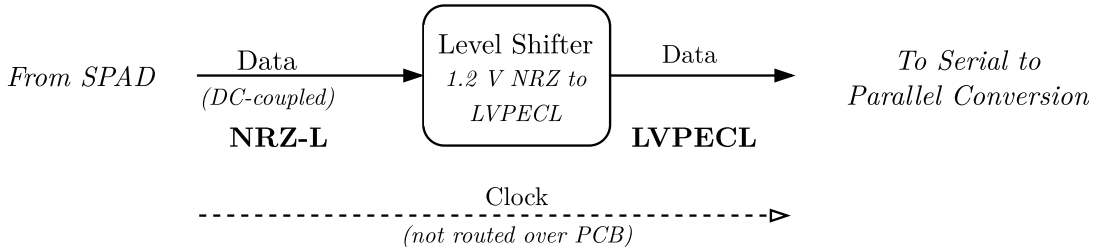


Figure 3.21.: Block diagram of the Level Shifter PCB. Bold uppercase text specifies the used voltage line levels.

The structure of the Comparator PCB can be seen in fig. 3.18. Because the custom comparator IC has no clock output, the clock needs to be split using a 1:2 fanout buffer. One of the clock outputs goes to the comparator IC via a bias tee (to adjust the bias level). The other goes to the output SMA port of the PCB.

Most of the board space is dedicated to the custom comparator chip. Board power ($V_{CC} = 3.3\,\mathrm{V}$) is supplied via the JST-XH connector, which in turn is connected to either the Shift Register or Demux PCB. Since the comparator IC runs on 1.2 V LDO regulators with fixed 1.2 V output voltage[20] are used to supply that voltage. To minimize the effects of power rail noise, each kind of supply pin (V_{DD}, V_{Co}, and $V_{DD,\mathrm{Inv}}$) has its own 1.2 V LDO. In addition, a fourth 1.2 V LDO supplies the various tuning inputs (NA, NL, PM, DEL) via individual potentiometers. These each work as voltage dividers and allow to adjust the tuning voltages. The tuning inputs allow for the adjustment of things like MOSFET gate voltages (PM), voltage levels of internal clock signals (NA,NL) or delay line drivers (DEL).

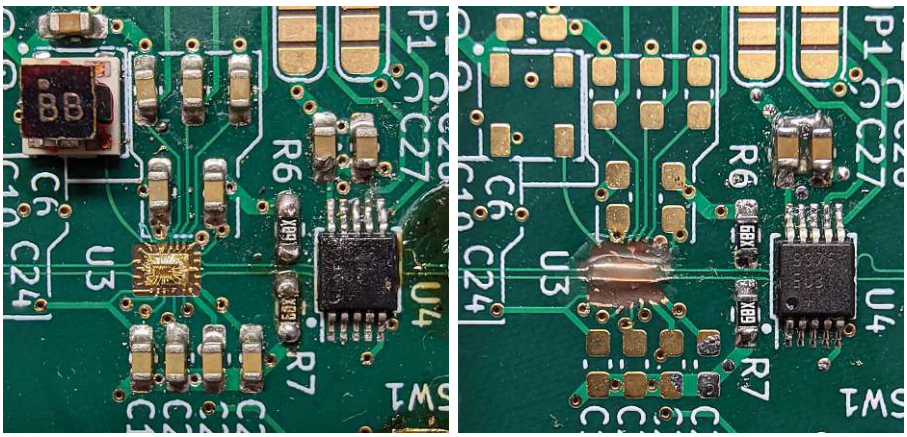
The PCB has a size of 50 by 100 mm. It can be seen in figs. 3.19 and 3.20. The comparator IC is directly glued and wire-bonded to the PCB.

3.3.5. Level Shifter PCB

The Level Shifter PCB is an alternatively assembled variant of the Comparator PCB. In cases where only the level shifting is needed and the comparator can be omitted this is the preferred module, as the custom comparator needs careful tuning to work as intended. The level shifter IC can also handle the small pulse amplitude levels of SPAD and APD receivers, having an minimum differential input voltage threshold of 100 mV. The simplified block diagram can be seen in fig. 3.21. Note that the clock signal in this variant is *not* routed over the PCB but the cables carrying the clock signal are directly connected to the (AC-coupled) Shift Register or Demux PCB.

Since the same PCB is used, it is required to bypass the comparator footprint, because it sits between the data input SMA connectors and the input pins of the level shifter IC. To bypass the footprint, the GND pad was ground off in the area of the footprint. Then, data input and output traces were connected via 0.2 mm soldered wires. Finally, the wires were covered with solder resist to electrically isolate

[20] AZ1117CH-1.2



(a) Comparator PCB (b) Level Shifter PCB

Figure 3.22.: Comparison of assembled components between Level Shifter PCB and Comparator PCB. The comparator IC can be seen near the centre of the image in (a). The Level Shifter PCB utilizes the same PCB but bypasses the comparator IC footprint through manual modification.

and also mechanically secure them (see fig. 3.22).
This leaves a mostly unpopulated PCB that really only contains the level Shifter IC, its accompanying circuitry and SMA as well as the JST-XH power connector.

3.3.6. Complete Assembly

The individual PCBs have to be connected to each other not just electrically, but also mechanically. As mentioned previously, the FX3 SuperSpeed Explorer Kit stacks on top of either the Shift Register or Demux PCB using standard 2.54 mm pin headers. Due to their high pin count they provide plenty of mechanical strength, so no additional components (e.g. screws and fasteners).

The Shift Register PCB and Demux PCB have 3D-printed carriers, that are essentially just several legs joined by beams below the PCB. The PCBs are screwed into the carrier using M3 screws. In the case of the Shift Register PCB there is also a mounting point for a 40 mm fan to provide active cooling.

The Comparator and Level Shifter PCBs use the same carrier. It mostly functions as legs with some mechanical bracing. However two ‘arms’ protrude to one side, allowing to mechanically connect the PCBs to either the Shift Register or Demux PCB carriers using M3 screws from the bottom side. The arms contain long slots, allowing to adjust the distance between the PCBs (see fig. 3.23).

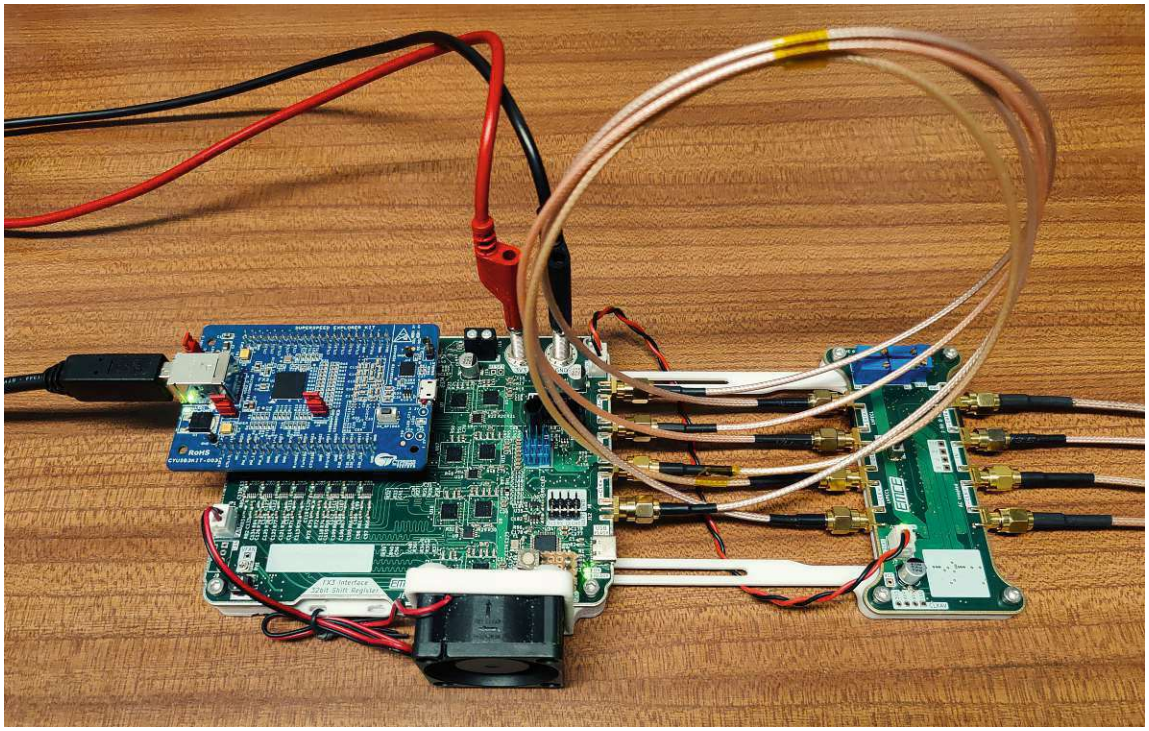


Figure 3.23.: The completely assembled interface with the FX3 SuperSpeed Explorer Kit on the left on top of the Shift Register PCB and the Comparator PCB to the right.

4. Results

All modules were evaluated for their transfer performance and power usage. Measurement setups are described in appendix A on page 61. Where possible, each module was measured independently, to guarantee that any incorrectly transferred bits can be attributed to the correct module. This was also essential in debugging and verifying basic function in the development stage. First, each module's transfer performance (achievable bit rates) and reliability (bit errors) will be shown. Then, power consumption and thermal distributions will be analyzed.

In cases where measurement issues, unexpected, and/or non-ideal performance occurred, this will be pointed out. This chapter will focus on the results of the operation. Furthermore, discussion of the reasons for failures with possible workarounds will be presented in chapter chapter 5 on page 45.

4.1. Performance Metrics

One of the most important metrics for performance of high-speed serial links is the *bit error ratio* (BER). It is defined [39] as:

$$\text{bit error ratio} = \frac{\text{number of bit errors}}{\text{total number of bits transferred}} \tag{4.1}$$

An ideal digital communication system would have a BER of 0 whereas a perfectly unreliable system would have a BER of 0.5 ($= 5 \times 10^{-1}$), indicating a completely random transmitted sequence.[1] In real systems, the BER is adversely affected by factors such as noise, interference (e.g. crosstalk) and distortion.

A closely related and also often used metric to the bit error ratio is the *bit error rate*, unfortunately also abbreviated as BER[2]. It is defined as the bit error ratio multiplied by the bit rate. The result is the number of bit errors per unit of time. While the bit error ratio is a *relative* measure (how many bit errors per transferred bit), the bit error rate is an *absolute* measure (number of bit errors per unit of time).

Since the bit error rate scales with the bit rate and the modules are evaluated for a wide range of bit rates, this thesis only uses the bit error ratio as defined in eq. (4.1). This allows directly comparing BER measurements for different bit rates.

[1] This is due to each bit having only two possible states. If the BER would be 0.7 for example, the transmitted binary sequence could be inverted and the BER would improve to 0.3.

[2] Confusingly, 'bit error rate' is sometimes used when the bit error ratio is actually meant. Always check the definition if available.

Sequence Name	Length of LFSR	Sequence Length (bits)	Sequence Length	Max number of consecutive zeros
PRBS7	7	127	127 bit	6
PRBS15	15	32,767	32.77 Kbit	14
PRBS23	23	8,388,607	8.39 Mbit	22
PRBS31	31	2,147,483,647	2.15 Gbit	30

Table 4.1.: Commonly used PRBS sequences for testing serial interfaces.

Measuring the bit error ratio

To test the reliability of a serial link and measure the BER, a known test pattern needs to be transmitted. This pattern needs to have a similar characteristic to a random sequence, to guarantee various difficult transfer situations that stress the interface (e.g. sequences of quickly alternating 1s and 0s or long sequences without a logic state change). It also needs to be fully deterministic, so its possible to check for the presence of bit errors in the received bitstream (by knowing/generating the ideally transferred sequence).

A commonly used pattern that fulfils these goals is the class of pseudo-random binary sequences (PRBS) [40, p. 2]. These are generated by standard deterministic logic elements, most commonly in a structure called linear feedback shift register (LFSR) [17, pp. 975-977], [40, p. 4]. Depending on the size of the LFSR, these patterns vary greatly in size, rising exponentially in powers of two. According to CCITT recommendations [41], [42], a PRBS pattern consists of $2^n - 1$ bits, with n being the length of the LFSR. The longest string of consecutive 0s equals $n - 1$. Thus, longer PRBS sequences stress the system more by having more variation in bit patterns. However, they also take longer to (completely) transmit. Table 4.1 shows the length of commonly used PRBS sequences.

4.1.1. Comparator PCB

The Comparator PCB has the job of guaranteeing clean, sharp transitions between logic states and shifting voltage levels to LVPECL levels (as described in section 3.3.4 on page 29). BER measurements for different PRBS patterns and transfer speeds are shown in fig. 4.1. PRBS7 patterns work without issues up to 3000 Mbit/s, with an increase in BER to 2.72×10^{-7} after that. PRBS15 works flawlessly up to 2000 Mbit/s, rising to a BER of 6.33×10^{-9}. PRBS31 only works up to 2000 Mbit/s (see fig. 4.1).

Unfortunately, there were significant difficulties in measuring the BER. For higher speeds (especially with harder PRBS patterns) it was not possible to achieve a sync in the used bit pattern receiver (i.e. the pattern receiver couldn't properly sample the incoming data with the incoming clock). Without a sync, no BER measurement could be done. This is the reason that data points at higher speeds are missing for

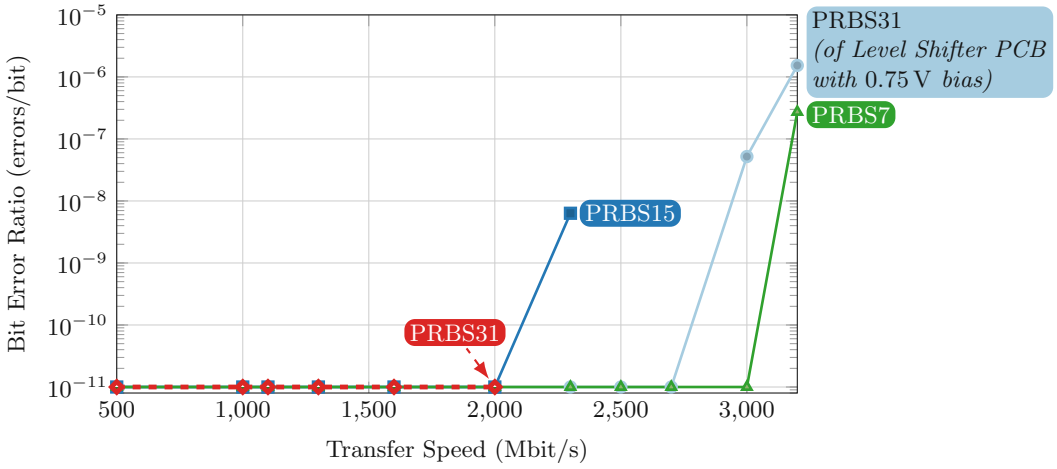


Figure 4.1.: Transfer reliability for the Comparator PCB and different PRBS test patterns. Most fitting Level Shifter PCB values added for comparison. 1×10^{-11} errors/bit is the measurement limit, i.e. data points at this value represent zero measured errors.

the PRBS15 and PRBS31 patterns in fig. 4.1. These difficulties will be discussed in more detail in section 5.2.1 on page 46.

4.1.2. Level Shifter PCB

If only level shifting is required, the Level Shifter PCB is an alternative to the Comparator PCB that omits the comparator (as described in section 3.3.5 on page 31). Its only purpose is shifting a differential data signal to LVPECL voltage levels.

According to the datasheet the level shifter IC can 'accommodate any differential signal within rails'. The input common-mode voltage can reach from 0.05 V to $V_{CC} - 0.05\,\text{V}$, the differential input threshold voltage is as low as 100 mV. The switching frequency is guaranteed to at least 2 GHz, with the typical value at 2.5 GHz and no stated maximum.

In the measurements, the achievable BER was strongly dependant on input common-mode voltage. If the voltage is close to the rails, the chip handles the advertised speeds and more. Between 0.75 V and 1.75 V the bit error ratio increases strongly, with 1.25 V being nearly unusable at speeds above 1500 Mbit/s, having BERs of 1×10^{-3} and worse. The BER values are also wildly fluctuating at those speeds (with 1.25 V) See fig. 4.2.

4.1.3. Shift Register PCB

One of the two variants to handle serial-to-parallel conversion is the Shift Register PCB (as described in section 3.3.2 on page 22). This PCB uses daisy-chained shift registers to handle the conversion.

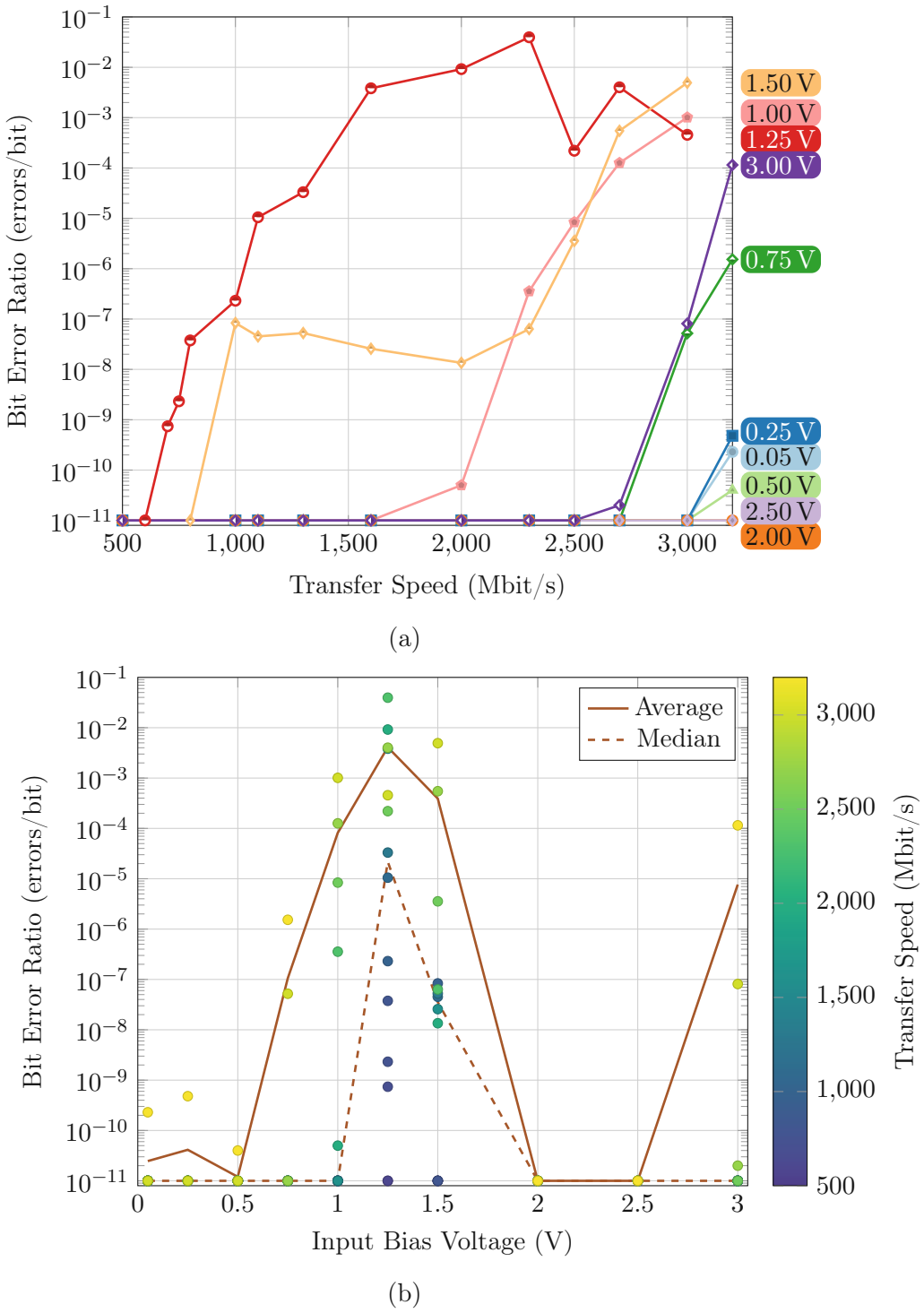


Figure 4.2.: Transfer reliability for the Level Shifter PCB and a PRBS31 test sequence. Performance is strongly dependant on common-mode input voltage, with a strong increase in bit errors in the range of 1 V to 1.5 V. 1×10^{-11} errors/bit is the measurement limit, i.e. data points at this value represent zero measured errors. The data points represent BER measurements at different transfer speeds and input voltages. Measurements for PRBS23 are shown in the Appendix in fig. C.3 on page 69 but demonstrate roughly the same reliability.

The BER measurements (fig. 4.3) show a complicated picture: Below 1330 Mbit/s (apart from three unusable[3] bands of about 80 Mbit/s each), the PCB is functional, which means that there were no measured bit errors. Then a larger unusable band occurs up to 1480 Mbit/s.

Between 1480 Mbit/s and 2040 Mbit/s (with an unusable band at 1600 Mbit/s to 1710 Mbit/s) the PCB is semi-functional. It can transfer data without measured bit errors, but every 9th bit is lost due to a systematic issue that will be explored in detail in chapter 5. This makes it unusable for actual data transfer, but since SPAD characterization measurements are stochastic in nature, it could still be used in specific cases. A more detailed explanation as well as two ways to compensate for this in PRBS testing will follow in section 5.2.2 on page 48.

4.1.4. Demux PCB

The alternative to the Shift Register PCB is the Demux PCB. It handles serial-to-parallel conversion using cascaded demultiplexers and deserializers (as described in section 3.3.3 on page 26).

Compared to the Shift Register PCB it allows for much higher transfer speeds (up to 3310 Mbit/s). There is an unusable band at 1480 Mbit/s to 1670 Mbit/s, but all other speeds up to the maximum are usable (see fig. 4.3).

As explained in section 3.3.2 on page 22, specifically in fig. 3.15, the output bits are systematically at the wrong bit positions within each 32-bit word and need to be corrected in software after measurement.

There is another issue: randomly, the output gets 'doubled', i.e. the same 32-bit word is repeated once (or inserted an additional time). After that the rest of the PRBS sequence continues as normal, but shifted by 32-bits compared to before the word repetition. This can happen as early as after transferring 10 Mbit or as late as after several transferred Gbit but most often in the range of several hundred Mbit (allowing to measure BERs of up to 1×10^{-8} or better in one continuous measurement). This can be detected algorithmically and potentially automatically compensated for but needs to be taken into account when using it for measurements.

These issues (and some possible solutions) will be explored in section 5.2.4 on page 50.

4.1.5. FX3

The FX3 handles USB communication and transfers to the PC. It is usable up to a speed of 3050 Mbit/s (see fig. 4.3), slightly exceeding the maximum speed of 3000 Mbit/s given in the datasheet [24, p. 8], although speeds of up to 3586 Mbit/s have been reported [28, p. 9]. In our case, above 3050 Mbit/s, the FX3 immediately locks up and becomes unresponsive after starting a transfer, requiring a hard reset.

[3] Unusable meaning that the BER was either all the time or most of the time worse than 1×10^{-2}.

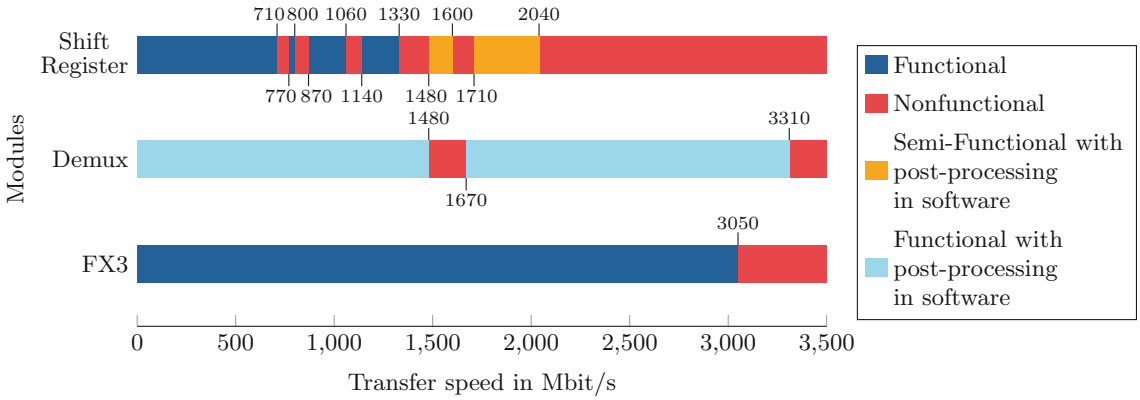


Figure 4.3.: Comparison of usable frequency ranges.

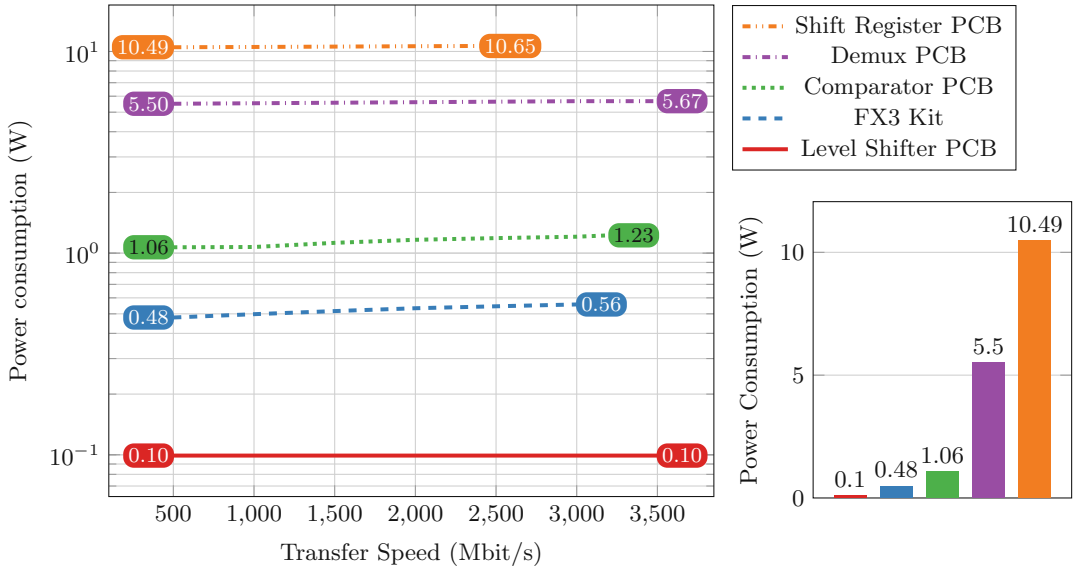
The FX3 itself introduces no bit errors but has an issue with data integrity. Randomly, 16.384 kB will get lost (i.e. they are missing from the transferred data). It occurs after 1 Gbit of the transmitted sequence on average (still allowing to measure BERs of 1×10^{-9} or better in one continuous measurement). Outliers can go as low as 10 Mbit or as high as several Gbit. 16.384 kB is the size of one DMA buffer. The reason for this issue will be explored in section 5.2.5 on page 52.

4.2. Power & Thermal Analysis

The power consumption of all PCBs was measured for various transfer speeds (see fig. 4.4). As mentioned in section 3.3.1 on page 18, an important factor when using LVPECL logic levels is the number of terminations, since power consumption scales with it. The Demux PCB has 30 LVPECL terminations (counting differential pairs as two terminations since they are driven by two ECL drivers). The Shift Register PCB has 140 LVPECL terminations. As expected, the Shift Register PCB has the highest power consumption, surpassing 10 W (see fig. 4.4b).

Power hardly rises with frequency on the ECL-dominated PCBs (Level Shifter PCB, Shift Register PCB, and Demux PCB). This is expected, as the ECL logic family does not have much additional power loss while switching compared to other logic families like TTL or CMOS [43], at the cost of much higher quiescent power consumption. Active dissipation in TTL can be up to 25 times higher than in ECL when driving identical load capacitors [44, pp. 45-46]. The Comparator PCB and FX3 are dominated by CMOS technology and thus show a noticeable increase in power consumption with higher transfer speeds (see fig. 4.4a).

Given the power consumption of the Demux PCB with approximately 5.5 W and the Shift Register PCB with approximately 10.5 W, thermal considerations needed to be taken into account as well. Both PCBs produce noticeable heat during operation. The Demux PCB can still be passively cooled, peaking at 62.2 °C and spreading the heat over a large area (see fig. 4.8). Care should be taken to ensure proper airflow in



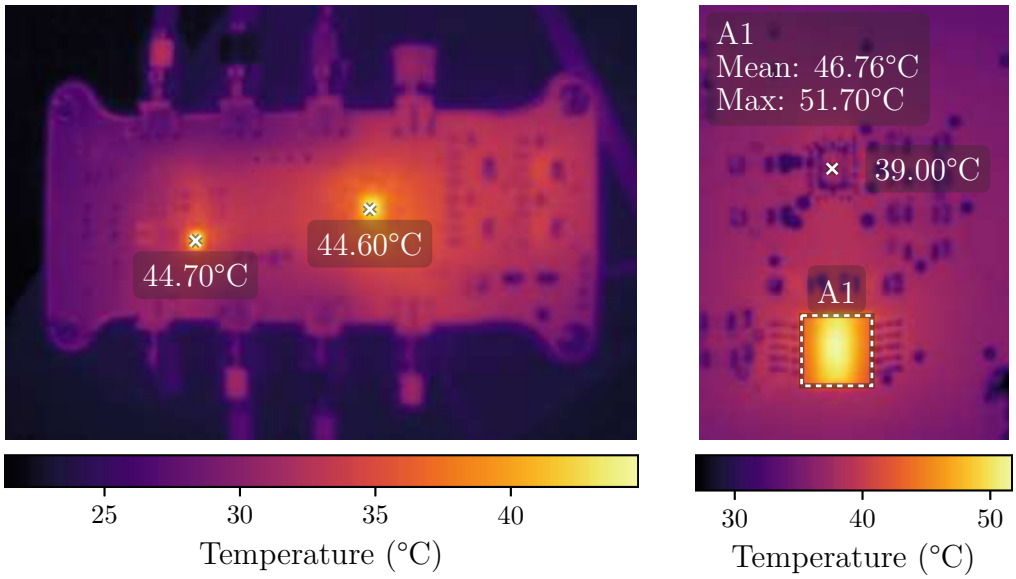
(a) Logarithmic scale (b) Linear scale (at 500 Mbit/s)

Figure 4.4.: Power consumption of the different modules. Note how the CMOS-dominated PCBs (Comparator PCB & FX3 Kit) rise in power consumption with higher transfer speeds. The ECL-dominated PCBs (Shift Register PCB, Demux PCB, Level Shifter PCB) hardly rise at all with transfer speed.

case it will be enclosed.

The Shift Register PCB on the other hand both produces significantly more heat and has that heat concentrated in a smaller area. This necessitates active cooling using a 40 mm fan blowing over the PCB from the side. Using this, temperatures peaked at 56.3 °C (see fig. 4.7). Without active cooling, temperatures rose above 95 °C, exceeding the maximum operating temperature of the shift register ICs of 85 °C.

The Comparator PCB stayed under 45 °C even when passively cooled (see fig. 4.5) and the FX3 stayed at a cool 32.6 °C.



(a) (b)

Figure 4.5.: Thermal image of the Comparator PCB. (a) Whole PCB. (b) Close-up of the left hotspot of (a), with the level shifter IC on the bottom and the comparator IC on top. For reference, a rendering of the corresponding PCB is shown in fig. 3.19 on page 30.

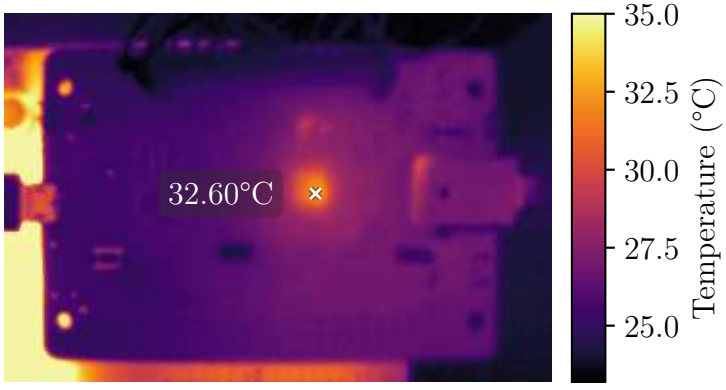
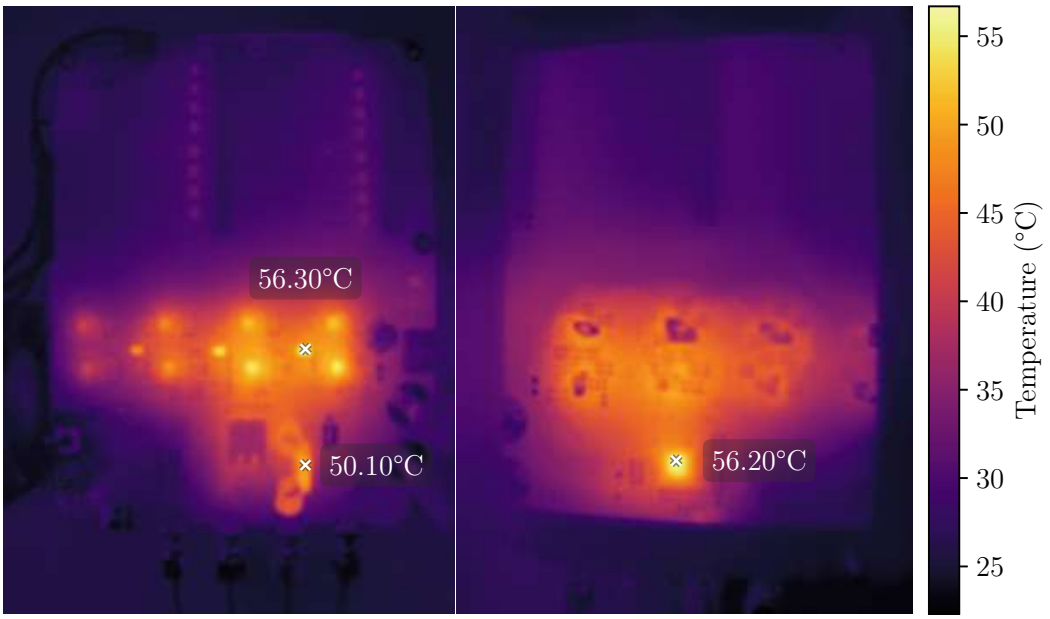
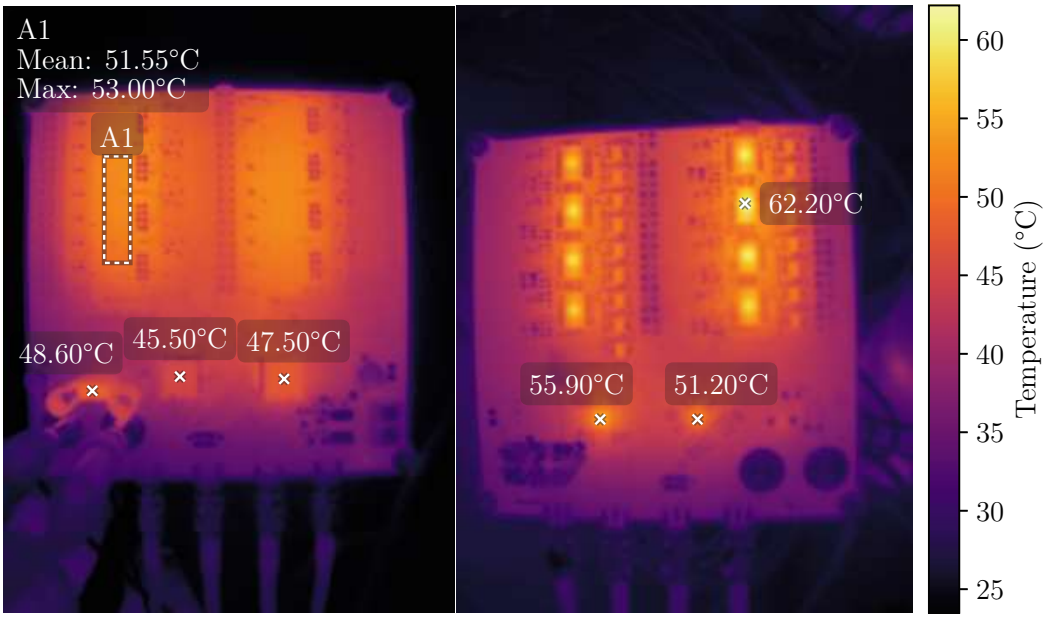


Figure 4.6.: Thermal image of the FX3 SuperSpeed Explorer Kit. The much warmer Shift Register PCB can be seen in the background to the left.



(a) Top (b) Bottom

Figure 4.7.: Thermal image of the Shift Register PCB (without FX3 stacked on top). A cooling fan blows air from the left in (a), creating the temperature gradient. For reference, a rendering of the corresponding PCB is shown in fig. 3.12 on page 25.



(a) Top (b) Bottom

Figure 4.8.: Thermal image of the Demux PCB (without FX3 stacked on top). For reference, a rendering of the corresponding PCB is shown in fig. 3.16 on page 28.

4.3. Summary

- Bit error ratio (BER) is an important metric for characterizing the transfer reliability of high-speed serial links. It is defined as the *number of bit errors* divided by the *total number of bits transferred*.
- BER is commonly measured using pseudo-random binary sequences (PRBS). They exist in various sizes, with larger patterns introducing more 'randomness', thus stressing the link more.
- The Comparator PCB works over the whole transfer speed range with a PRBS7 pattern but can only reach 2000 Mbit/s with the much harder PRBS31 pattern. This is below expected performance.
- The alternative Level Shifter PCB can cover the whole transfer speed range and allows nearly any differential input signal within rails. Contrary to the datasheet, performance drops drastically at input common mode voltages from 1.00 V to 1.50 V, with BER rising above usable levels at 1.25 V above 1000 Mbit/s.
- The Shift Register PCB works up to 1130 Mbit/s but care needs to be taken to choose a working transfer speed since some frequency bands do not work. For characterization measurements of SPADs/APDs that can deal with every 8th bit missing it is usable up to 2040 Mbit/s.
- The alternative Demux PCB works up to 3310 Mbit/s but needs post-processing in software to correct bit placement. It also has the issue of randomly repeating 32-bit words, typically occurring after several hundred Mbit but in some cases as early as 10 Mbit or as late as several Gbit.
- The FX3 works up to 3050 Mbit/s but the transferred data is randomly missing $\approx$ 16 kB, the size of a DMA buffer. This happens mostly after at least 1 GB, but can occur sooner.
- Explanations and workarounds to the aforementioned issues are explored in the next section.
- A visual comparison of the usable transfer speed ranges is shown in fig. 4.3 on page 40.
- Power consumption scales with the number of PECL terminations but hardly over frequency in PECL-dominated circuits (Level Shifter, Shift Register, and Demux PCBs). The Shift Register PCB needs $\approx$ 11 W, the Demux PCB $\approx$ 6 W. The other PCBs stay at or under $\approx$ 1 W.
- Waste heat produced by the Shift Register PCB is high enough to require active cooling with a 40 mm fan. All other PCBs can be passively cooled.

5. Discussion

Every electronic system has limitations in performance, some more, some less. This section will focus on things that went wrong, perform below expectations or suffer from notable shortcomings. Design decisions will be discussed and possible causes for issues as well as workarounds will be explored.

5.1. System Architecture

An essential decision to make at the beginning of the thesis was about the fundamental approach to handle the serial-to-parallel-conversion (see section 3.1 on page 7). Two potential strategies were considered: implement an **FPGA-based** solution, or make the conversion using **discrete components**. *Perceived* advantages of the latter were:

1. **Easier troubleshooting:** Any issues before entering the FX3 would be strictly confined to hardware. No need to debug problems caused by bugs or mistakes in the hardware description language (HDL) of the FPGA.

2. **Easier assembly and rework:** Nearly all FPGAs come in ball-grid array (BGA) packages, with their pins below the chip. In case of bad solder jobs (all boards were assembled by hand) or even just when reworking the board or switching to a new revision of the PCB, the BGA chip would need to be reballed. No in-house tools or expertise are available for a possible re-balling, so an external service provider would need to be used.

3. **Shorter design time:** As no HDL code has to be written, simulated and debugged, there would be less time needed to get out the (first) design and evaluate if the approach is working or not. On a functional level and compared to FPGAs, discrete components are small, easy to understand modular blocks that can be verified (mostly) independently of others.

4. **Better availability:** At the time of the decision, there was a global supply chain crisis, with much of the FPGA selection being out of stock and lead times between months and years. The few available ones were usually high-grade variants with unit costs of several hundred euros.

5. **Better pricing:** In addition to the limited availability mentioned in the item above, custom FPGA hardware designs require a paid license for the FPGA design software. This can be in the range of thousands of Euros per year.

Many of these assumptions would turn out to be wrong or at least it is highly questionable that an FPGA-based approach would have fared much worse:

1. **Easier troubleshooting:** While it is impossible to say how much work troubleshooting an FPGA-based design would have been, the unfamiliarity with LVPECL logic and some unwelcome surprises in the form of undocumented behaviours of ICs (more on this later) led to months of troubleshooting, (re-)measuring and board revisions.

2. **Easier assembly and rework:** This held true.

3. **Shorter (hardware) design time:** The large number of transmission lines made routing on a four-layer board challenging and time-consuming.

4. **Better availability:** Discrete components were also hit by the supply chain crisis. In addition, ICs handling such high speeds are quite specialized, with only a small selection fit for the application. In practice, this actually made selecting parts harder, because these chips were challenging and time-consuming to find in the first place. Many other chips did not fit the application but could not be filtered out at distributor or manufacturer websites, so dozens of datasheets had to be read.

5. **Better pricing:** Because of the specialized nature of the discrete ICs their unit costs were in the range of 5 to 20 € each. With several of these chips needed for each design, costs quickly rose to what available FPGAs would have cost. However, the yearly license fees could be completely avoided with the discrete-approach.

Due to the perceived advantages at the time it was decided to go with the discrete approach. The idea was to quickly get a design working and if it should turn out to be unreasonable or impossible, the FPGA-based design could still be pursued. In hindsight, it would probably have been a good idea to do just that early, before addressing all of the issues encountered with the discrete designs.

But due to the designs often being seemingly just one 'fix' away from doing their job as well as the already invested time and effort this point or decision was never reached until it was too late to be reasonable within the scope of this thesis.

5.2. Performance Analysis & Limitations

As in the previous chapters, each module will be looked at separately.

5.2.1. Comparator PCB

As mentioned in section 4.1.1 on page 36, BER values for speeds above 2000 Mbit/s for PRBS31 and above 2300 Mbit/s for PRBS15 could not be measured.

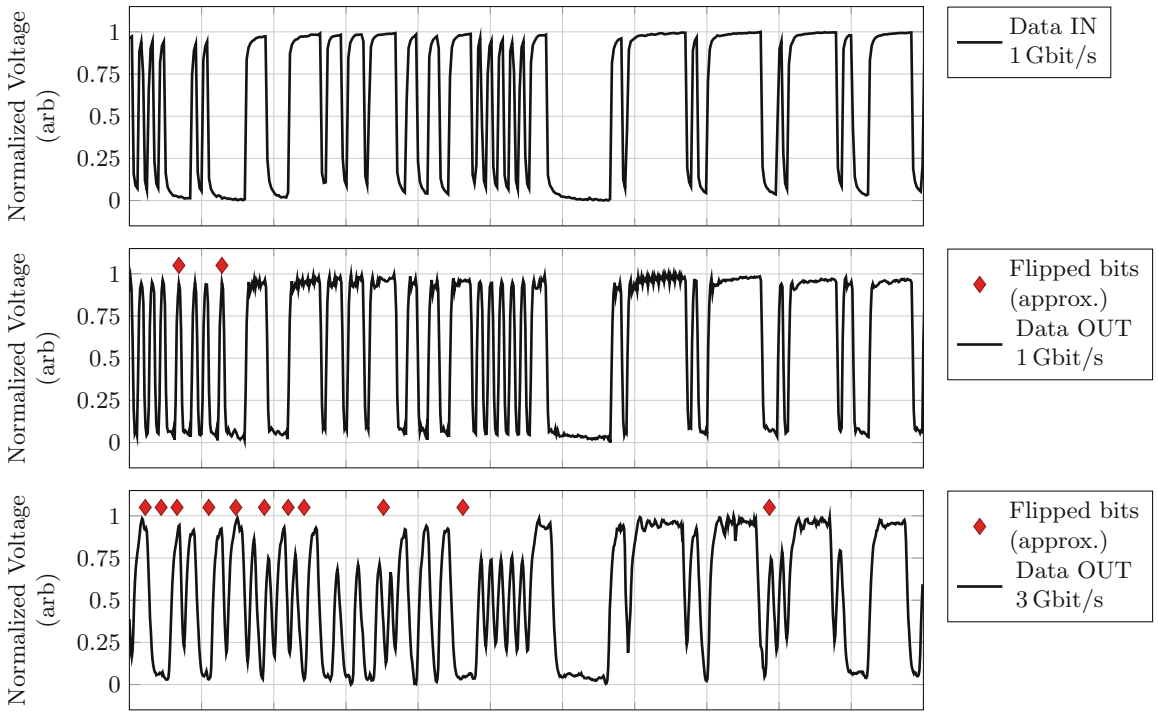


Figure 5.1.: Qualitative comparison of data input signal with data output signal of Comparator PCB at two different speeds. Locations of flipped bits are (roughly) marked. Note how the limited slew rate at 3 Gbit/s prevents reliably crossing threshold levels in fast switching bit sequences. Flipped bits at 1 Gbit/s can be prevented with careful tuning but are included here for demonstration purposes.

The reason for these measurement difficulties and underperformance could not reliably be determined. One factor is that the slew rate is limited on the output (see fig. 5.1). The cause of this is unclear however, as the level shifter IC (which drives the output) can handle higher speeds, both according to the datasheet, as well as measurements (as shown in section 4.1.2 on page 37). The output driver of the custom comparator IC ought to be able to handle clock speeds of up to 7 GHz [38, p. 213], significantly surpassing the speeds used in this thesis.

In addition to the limited slew rate, there's also an increasing number of flipped bits in the data output at higher speeds, which could be attributed to any of the two ICs but should happen with neither to the experienced extent.

Another factor is phase shift between the data signal and sampling clock. If the shift is too large, the receiver will not be able to reliably sample the signal. This is likely the most important factor, as performance was highly sensitive to changes in phase shift.

Unrelated to these issues, the chip needs careful tuning of various parameters and auxiliary voltages to perform well. This allows it to adapt the PCB to many input

conditions but is detrimental to usability. Thus, it should only be used in cases where it is absolutely needed.

5.2.2. Level Shifter PCB

The BER of the level shifter IC is strongly dependent on input common-mode voltage (see section 4.1.2 on page 37). The cause of this dependence is not known. The manufacturer does not mention it in the datasheet at all. Instead, it is stated that nearly any common-mode voltage is allowed [45, pp. 1-2, 5], with no explicit restrictions on switching frequency.

A possible cause could be that a rail-to-rail input stage is used, with both n-MOS and p-MOS transistors. These can both be active simultaneously in the middle of the supply voltage range, leading to degraded performance in this region if not implemented well. Without knowing the internal makeup and topology of the chip, it is impossible to say, however.

Unfortunately, SPICE simulation to check for this behaviour is not possible either, as the manufacturer only supplies an IBIS model [46]. IBIS models only model the electrical properties of the input/output pins of the IC, the inner workings are not part of the model. Thus they only allow signal integrity simulations between two IC pins connected via transmission lines [47, p. 1].

Outside of the problematic input common-mode voltage ranges, however, the IC performs well, even exceeding specifications and is basically plug-and-play.

5.2.3. Shift Register PCB

This approach to serial-to-parallel conversion came with a lot of issues. As shown in section 4.1.3 on page 37, there are several unusable transfer speed bands spread across the bandwidth. This is because there are at least 11 critical paths regarding clock distribution and delays (see fig. 3.21 on page 31). The various shift registers and latches only have little tolerance for clock mismatch. The unusable bands likely occur at speeds where the non-ideally compensated delays lead to unstable outputs, most importantly, unreliable sampling of the latches.

Originally the plan was to individually configure clock delays with the programmable clock divider and fanout described in section 3.3.2 on page 22 since it also has built-in delay circuitry for each output channel. This was problematic due to two reasons: the divided clock (that is routed to the latches) can only be delayed in 8 steps of 131 ps each, which is rather coarse compared to the possible high-speed clock's period (down to ≈ 315 ps). More importantly, while the datasheet says the high-speed clock (that is routed to the shifters) can be delayed in 256 steps, it was overlooked that the step size is the (input) clock period. In other words, delaying the high-speed clock would only be possible for divided clock outputs, not the high-speed ones, which use the input clock directly.

Another large issue, which is not mentioned in the datasheet limits the top speed of the PCB to 1330 Mbit/s without workarounds. While the shift register ICs can

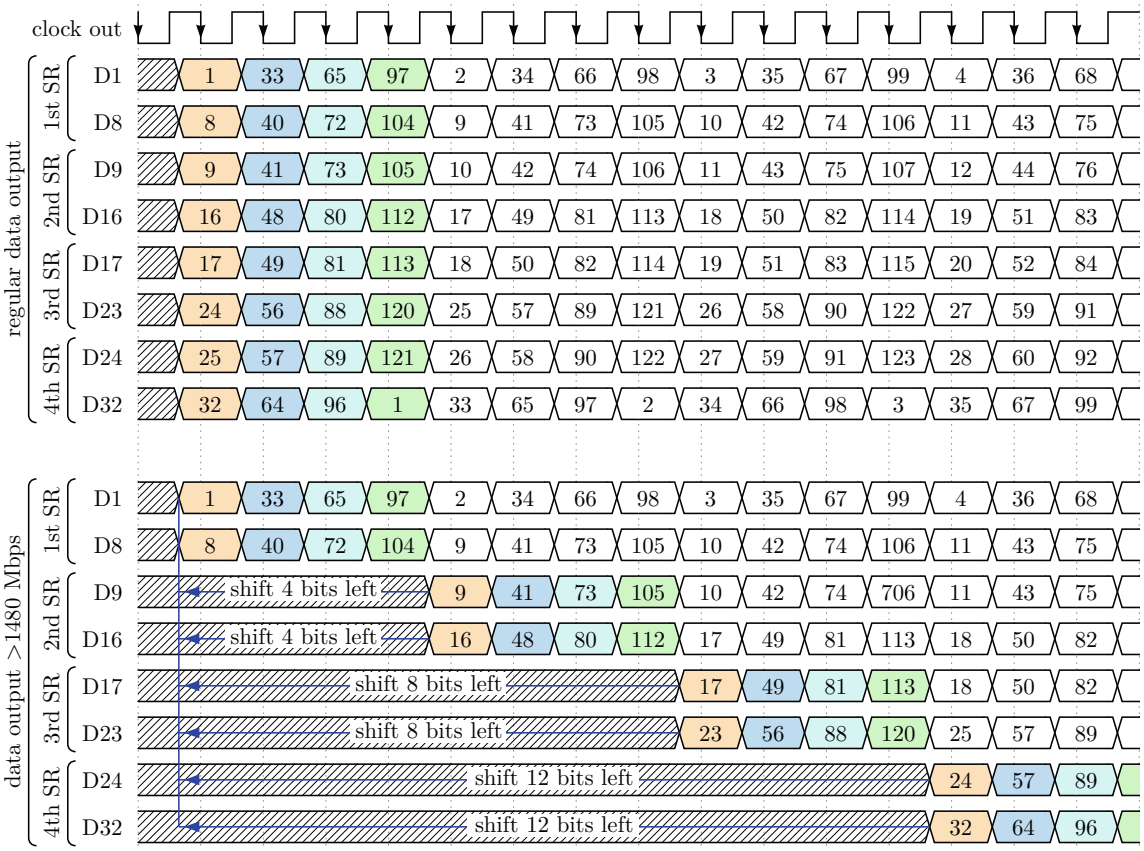


Figure 5.2.: Workaround for the Shift Register PCB to make BER measurements using a PRBS7 sequence. Bit numbers denote the position within the PRBS7 sequence starting with 1, ending with 127 and repeating after that. Rows show the first and last outputs of each shift register. The upper part shows the regular output and the lower part shows the output above the critical speed as well as the needed correction for PRBS7.

handle speeds up to 5 GHz, this only applies to *a single* IC. With four 8-bit shift register ICs daisy-chained together to get to 32-bit shift width, the input-to-output propagation delay needs to be taken into account as well. The shift register ICs add 550 ps–800 ps of delay, a 1:2 fanout (with one output going to the FX3 and one to the input of the next shift register IC) adds 155 ps–265 ps and the PCB traces add another ≈ 90 ps, giving a total of 795 ps–1155 ps. The period of a 1000 Mbit/s signal is 1000 ps which is in that range.

This means that these delays surpass the periods of higher bit rates used in this application. Above a bit rate of 1480 Mbit/s an issue starts to appear: the last output of the shift register generated on one clock edge reaches the second shift register IC during a clock edge *after* the next clock edge. In other words, there is an unwanted delay of one or more clock cycles. The effect of this is that above the critical speed, the outputs of the second shift register IC are ‘shifted’. Example: Imagine a serial

input bit word of 32 bits, with each bit numbered from 1-32. Ideally the parallel output word would be bits 1-8 for the first shift register, 9-16 for the second shift register, and so on. Above the critical speed instead you would get: 1-8, 8-15, 15-22, and 22-29. Bits 30-32 are now missing and the outputs of shift registers 2, 3 and 4 are shifted by one bit. This repeats for the next 32 input bits and so on. So the output of the first shift register is always 'correct', while the outputs of the second to fourth shift registers are shifted and also missing one bit each. This behaviour can also be reproduced in simulation.

There is a way to work around this however, at least for BER measurements using PRBS sequences. Since PRBS sequences always have a length of $2^n - 1$, which is not divisible by 32, it is guaranteed that all parts of a PRBS sequence will occur within the output of each shift register eventually. It is possible to construct the correct PRBS sequence by shifting the output of the 'wrong' shift registers depending on the used PRBS pattern (fig. 5.2). Note that this still means that every 30th to 32nd bit is missing, we just manipulated the output in such a way, that a continuous measured PRBS sequence is achieved. Since the BER measurement usually is stochastic in nature, with bit errors occurring randomly, this should still give correct results in those scenarios. If, however, the device under test had some error that systematically flips every nth bit or similar, this workaround could distort the resulting BER and should be avoided.

Another workaround would be to remove every 30th–32nd bit from the ideal PRBS sequence the measurement is compared to and removing every 8th, 15th, and 22nd bit from the measurement. This may be easier than determining the correct shift offsets of the above method but comes with the same limitations.

5.2.4. Demux PCB

While the Demux PCB works much better than the Shift Register PCB, it still has an unusable band between 1480 Mbit/s to 1670 Mbit/s. This is likely due to not perfectly compensated delays. Since the Demux PCB is much simpler and the ICs divide the clock themselves, clock mismatch is a much smaller issue, however, leading to only one unusable band.

Regarding transfer speed, the upper limit exceeds the 3200 Mbit/s the FX3 can handle. There is the 'cascaded demultiplexer problem' to consider, as explained in section 3.3.3 on page 26, specifically fig. 3.15. This is not a performance issue, it just makes evaluation in software after measurement more complicated.

A genuine problem, however, is the problem with 'doubling' output words, i.e. the same 32-bit word is repeated once (or inserted additionally). A possible explanation for this would be if the synchronous clock fed to the GPIF II interface of the FX3 had an errant double pulse, leading to it sampling the input twice. However, it was not possible to observe or trigger this behaviour even once using an oscilloscope.

A possible workaround in software exists for BER measurements using PRBS sequences. The idea would be to detect if a 32-bit chunk of the measurement repeats and if so, remove it. In an ideal PRBS sequence, it is impossible for a 32-bit sequence

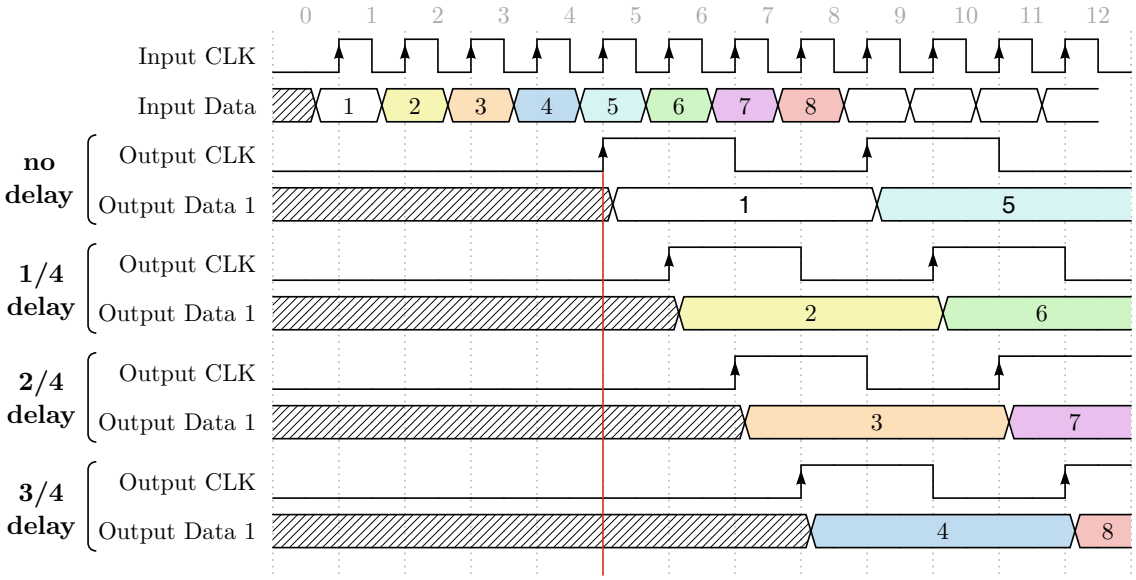


Figure 5.3.: Four possible startup behaviours for a 1:4 deserializer. These differ in startup time, thereby changing which input bit is considered the first. Edges on the outputs shift accordingly. The first bit is output to the first output pin of the deserializer (Output Data 1), then the second (Output Data 2, not shown), and so on. Bit No. 5 is then output to the first output pin again. A more comprehensive variant of this figure with all outputs is shown in fig. C.2 on page 68.

to repeat (within one sequence). With enough bit errors at the wrong locations, it would be possible, although unlikely, so this workaround isn't perfectly robust.

A more robust way of detection would be to calculate the BER in chunks (by comparing the measurement with the ideal sequence piece by piece). If BER suddenly jumps to $\approx 5 \times 10^{-1}$, somewhere within the observed chunk a doubling error occurs. Detecting the error opens the way to algorithmically remove it.

There exists another unforeseen limitation which does not affect performance: The eight 1:4 deserializer ICs that are cascaded after the 1:8 demultiplexer have random startup times. Because of this, each deserializer misses varying amounts of bits *before* startup. *After* startup, the outputs are updated every fourth bit, with the first incoming bit mapped to the first output and so on. As a result, the eight deserializers start deserializing out of sync respectively to each other, having different bit orders and edge timings on the outputs (fig. 5.3). This behaviour is not mentioned in the datasheet.

The manufacturer built in a solution for this issue: there is a 'SYNC' pin that shifts the incoming bit which is considered the first by one bit. For example, this would change an initial '1/4 delay' state to the '2/4 delay' state in fig. 5.3. Unfortunately, this means that upon startup one needs to do a manual synchronization procedure, checking the output shift of each deserializer to the first one, and applying the correct

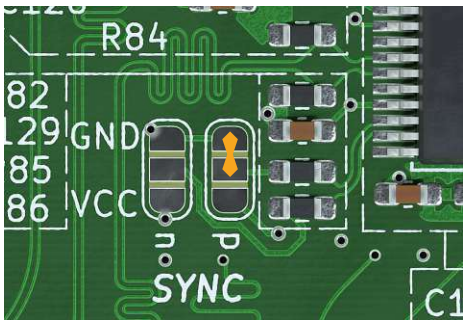


Figure 5.4.: The easiest way to send synchronization pulses to a deserializer is to briefly short the positive input SYNC pin with GND using a metal object. The relevant pads are marked.

number of SYNC pulses. Fortunately, a pulse or function generator is not necessarily needed for this, it is enough to simply short the positive SYNC input to GND with a metal object, e.g. the pin of a standard pin header or a small screwdriver (see fig. 5.4).

If the SYNC worked in a way that a pulse did not shift the output but 'reset' it, i.e. the first bit is the one that arrives after the pulse, then it could be used as a quick and easy global reset signal. Unfortunately this is not the way it works, so the explained manual procedure is needed, which adds several minutes of work after every startup. The startup states are random, it is not possible to send a fixed number of correcting pulses to each deserializer.

5.2.5. FX3

The FX3 performs slightly below expectations, falling short several percent of the maximum 3200 Mbit/s by locking up above 3050 Mbit/s. Below, it performs well at all speeds.

There is an important limitation: As mentioned in section 4.1.5 on page 39, randomly, 16 384 kB will get lost. This is the size of a single DMA buffer. It can happen that the USB host (i.e. the PC) is busy with other tasks and doesn't empty the buffers fast enough. The buffers fill up and the next incoming packet is dropped. Since the host operating system (Windows) is not a real time operating system (RTOS), it can't be guaranteed that the buffers are handled in time. Therefore, dropping of some data cannot be avoided.

Other factors contributing to dropped data due to full buffers, are:

1. **Shared bandwidth with other devices:** USB devices often share bandwidth with other USB devices, several ports may map to the same USB host controller. In addition, the host controller may be sharing traffic with other peripherals on the motherboard chipset's internal PCIe lanes to the CPU. Don't use USB hubs and – if available – check the motherboard's block diagram to figure out which USB ports on the rear of the PC are connected directly to the CPU.

2. **Storage medium too slow:** A hard disk drive (HDD) will likely be too slow for the higher transfer speeds. Use a solid-state drive (SSD). More speed is advantageous (with the SSD's cache also being a relevant factor). Benchmarks indicate however that all SSDs (connected via at least 6 Gbit/s SATA) can handle the maximum possible sustained write speeds.

3. **USB host controller:** The used USB host controller can have a considerable influence on the achievable throughput. Some host controllers are 20 % slower than others [28, pp. 13-14].

A solution to this would be to implement flow control. If the host's buffers fill up, a signal is sent to the FX3 to wait sending new data until a signal to resume transmission is sent. This reduces effective throughput (depending on the number of pauses) [48, p. 117] but guarantees that no data will be lost. This works for transfers like getting data to or from an external storage. However, the use case of this thesis is a streaming application; The device under test can't be told to wait. Data comes in continuously and the FX3's DMA memory is not large enough to buffer much data. In typical streaming applications like audio or video it is not a big issue if there is a small amount of data loss; It just leads to a momentary loss in service quality.

In this case it *is* an issue, but a minor one, because – as mentioned in section 4.1.5 – a BER of 1×10^{-9} and better can be measured in a single continuous measurement most of the time, which is lower than the typical BER of SPADs. Dropped buffers can also be detected with the same method explained in the above section: calculating BER in chunks and checking for a jump in BER to $\approx 5 \times 10^{-1}$.

It should be noted that dropped data due to full DMA buffers does not occur less frequently if transfer speed is reduced. In the range of 500 Mbit/s to 3000 Mbit/s there was no indication of any dependence of transfer speed to the amount of average transferred data until first data drop.

5.3. Improvement Suggestions

This section outlines some possible improvements for the given parts of the system.

5.3.1. System Architecture

Given the experiences of this thesis with the discrete approach, I would recommend pursuing an FPGA-based approach. The key requirements for an FPGA would be to have an integrated Serializer/Deserializer (SerDes) block that can handle input speeds of up to 3.2 GHz and enough digital outputs for 32 data lanes as well as a clock lane (all single-ended).

The discrete approach solutions can be enhanced further nonetheless, as explained in the next sections.

5.3.2. Comparator PCB

Unfortunately it could not be reliably determined what caused the underperformance of the Comparator PCB. Without knowing the cause, it is difficult to recommend any improvements. A start would be to isolate it from the Level Shifter IC, to be able to debug it properly. As mentioned in section 5.2.1, special care should be taken to minimize the clock and data phase shift, as this is likely the most important factor for degraded performance.

5.3.3. Level Shifter PCB

The only issue here was an unexpected dependence of the speed on common-mode input voltage for performance. This is an issue of the IC itself. This was (at the time of part selection) the only available off-the-shelf level-shifter IC that could handle both the speed as well as the 0.6 V common-mode voltage the comparator IC outputs.

5.3.4. Shift Register PCB

The most critical issue here is delay mismatches. Properly controlling of all delays would give a broader usable range of transfer speed. The large number of critical clock paths likely makes changes uneconomical though. The issue with the upper speed limit of 1330 Mbit/s remains, however and can't be solved as long as the shift registers are daisy-chained.

5.3.5. Demux PCB

Delay compensation could be improved upon to remove the unusable transfer band of 1480 Mbit/s to 1670 Mbit/s. The 'cascaded demultiplexer problem' cannot easily be solved in hardware, at least on four-layer PCBs.

Big improvements could be made to automate the startup synchronization procedure: due to the complexity, it will require a microcontroller and some analogue circuitry. It needs to be able to measure the phase offset of the output clocks of each 1:4 deserializer to the first, so 7 phase offsets. The signals that have to be compared can have a maximum speed of 100 MHz. If a phase offset exists, the corresponding number of pulses need to be sent to the respective deserializer (one pulse for 90°, two pulses for 180° and three pulses for 270°). The goal is to have no phase offset on all 8 deserializer clock outputs.

Another improvement would be part placement. The micro-USB port at the back of the FX3 (that gives access to the JTAG debugger as well as the UART debug console) is partly blocked by one of the heat sinks. It can still be used by pulling the FX3 halfway out of the pin header that connects it to the Demux PCB but this is not ideal of course. In normal usage, the debug port is not needed however.

5.3.6. FX3

There is the option of changing the DMA channels in the FX3 firmware from 'AUTO' operation to 'MANUAL'. This would allow the FX3 CPU to modify bits in the data stream, e.g. include information like input frequency, frame start and end sequences, or frame numbering to easily detect dropped data due to full buffers, etc. It would possibly also allow to correct the 'cascaded demultiplexer problem' shown in section 3.3.3 and fig. 3.15 before sending the data to the PC. However, 'MANUAL' DMA channels come at the cost of throughput, because the data can't be directly sent to the consumer socket by the DMA controller, instead being handled by the CPU [25, p. 80].

There is potentially also a way to detect buffer overflows from the FX3 side. When the FX3 buffers overflow (due to the host not emptying buffers quickly enough), a CYU3P_PIB_ERR_THRX_WR_OVERRUN event is triggered in the FX3 firmware [49]. While this can't prevent overflows, this could potentially be used to figure out the exact location in the streamed data and to send this information to the host.

5.3.7. Windows application

The windows application shown in section 3.2.1 on page 18 is perfectly functional, yet a bit bare-bones. Only the raw data is transferred, any associated metadata (transfer speed, measured device, type of measurement, measurement date/time, notes, etc.) must be recorded manually. It would be nice to add additional fields for these within the application's GUI and save these in an additional file next to the file with the transferred data. Suitable file formats would be YAML, JSON or CSV.

6. Conclusion & Outlook

The goal of this thesis was to develop and evaluate a USB interface capable of streaming incoming digital data from Single-Photon Avalanche Diodes (SPADs) or similar photodiodes to a PC via USB. This goal was achieved, albeit with some notable limitations.

For USB connectivity a *CYUSB3KIT-003 EZ-USB FX3 SuperSpeed explorer kit* [1] was used. For the interface, a topology of SPAD ⇒ comparator/level shifter ⇒ serial-to-parallel conversion ⇒ FX3 ⇒ PC was chosen. This involved designing custom four-layer PCBs for the comparator/level shifter and serial-to-parallel conversion stages. Each stage was implemented on its own PCB to aid debugging and modularity.

The comparator/level shifter was implemented in two configurations: one PCB integrated both a comparator and a level shifter, while the other featured only a level shifter. The serial-to-parallel conversion was achieved through two different topologies: one utilizing daisy-chained shift registers (Shift Register PCB) and the other using cascaded demultiplexers/deserializers (Demux PCB). Firmware for the FX3 was developed[1], including a simple state machine for the FX3's GPIF II interface. In addition, there is an easy to use windows application for initiating or stopping data transfers from the interface.

Results show that the whole interface is usable up to 3050 Mbit/s, approaching the theoretical maximum speed of the USB connection of 3200 Mbit/s. However, performance varied across modules. The Comparator PCB operated effectively with a PRBS7 pattern but was unable to meet the expected performance with the more complex PRBS31 pattern, achieving a maximum speed of 2000 Mbit/s. In contrast, the Level Shifter PCB supported transfer speeds of up to 3200 Mbit/s and any differential input within rails but experienced significant performance degradation at input common mode voltages between 1 V and 1.5 V.

The Shift Register PCB can be used up to 2040 Mbit/s but has every 8th bit missing from the streamed data above 1330 Mbit/s. The Demux PCB and FX3 can reach speeds of 3310 Mbit/s and 3050 Mbit/s, respectively, though both face issues with data integrity and missing bits.

A discrete component strategy was chosen over an FPGA-based one for the serial-to-parallel conversion stage due to perceived advantages such as easier troubleshooting, assembly, shorter design time, and reduced costs. However, many of these assumptions would prove to be misguided as the discrete designs led to significant challenges in troubleshooting, little flexibility and unsatisfactory performance results.

[1] Code can be accessed on a CD in the physical copy, online at github (`https://github.com/Fivefold/fx3_usb_interface`) or the Internet Archive (`https://archive.org/details/fx3_usb_interface`)

6.1. Comparison with high-speed oscilloscopes

Compared to high-speed oscilloscopes the USB interface has several distinct advantages:

Capture length: Oscilloscopes can capture data only up to their acquisition memory depth. Depending on the price of the oscilloscope this can range from ≈50 Mpts up to a few Gpts. If saving and transmitting the data to a PC is the objective, a new capture (beyond what fits into the acquisition memory) will not happen fast enough to achieve a continuous capture – at least on higher sample rates and acquisition depths[2].
The USB interface is not limited in this regard. Data can be continuously streamed (with the known data integrity issues) up to 3050 Mbit/s. Given a high enough input data speed, 10 or even hundred Gbit can be transferred to the PC within seconds.

Cost: While high-speed oscilloscopes with bandwidths of several GHz cost above 10 000 € and can surpass 100 000 €, the total cost of the interface is in the range of several hundred euros (depending on which module or configuration is chosen)

Usability: Oscilloscopes need careful setting up of triggers and various other parameters. The USB interface as a whole is mostly plug-and-play. With improvements (see section 6.3) it could become fully plug-and-play.

The biggest disadvantage of the USB interface compared to high-speed oscilloscopes is the severely degraded **data resolution**. While each sample of an oscilloscope has a resolution of 12 bit or more, the USB interface has a resolution of 1 bit, i.e. logic-high or logic-low. For SPAD and APD characterization purposes this is not necessarily an issue though. For many measurements, it is enough to measure the number and approximate width of pulses, the exact waveforms are not needed (for details refer to section 2.1 on page 3).

6.2. Module comparison & Usage Recommendations

Since three of the four modules come with relevant limitations, it is important to decide in which case to use which module. Table 6.1 gives a brief summary and comparison.

For the first part of the interface the choice is between using the Comparator PCB or Level Shifter PCB. Since the latter is much easier to use and also has better performance, it should be preferred over the Comparator PCB unless the differential

[2] There are ways to improve acquisition memory utilization on higher-end scopes if there are long idle times in the signal between relevant captures by cutting out the idle parts of the waveform. Keysight calls this 'segmented memory', Tektronix uses 'FastFrame'. However, this is still limited by acquisition memory depth and it is not a continuous capture.

Module	Comparator PCB	Level Shifter PCB
Performance	Up to at least 2000 Mbit/s*	Up to 3200 Mbit/s†
Ease of use	Careful tuning required	No special requirements, 'plug and play'
Limitations	No special limitations	No special limitations

*Depending on PRBS sequence, see fig. 4.1 on page 37.

†Depending on input common-mode voltage, see fig. 4.3 on page 40.

Module	Shift Register PCB	Demux PCB
Performance	Up to 1330 Mbit/s fully functional, up to 2040 Mbit/s semi-functional‡	Up to 3310 Mbit/s§
Ease of use	Turn on, apply valid input clock, press CAL button	Manual Synchronization procedure upon each startup (takes several minutes)
Limitations	None below 1330 Mbit/s, above that every 30th–32nd bit is missing	Randomly repeats a 32-bit word (after hundreds of Mbit on average)

‡Five narrow unusable speed bands over the whole range, see fig. 4.1 on page 37.

§Unusable band from 1480 Mbit/s to 1670 Mbit/s.

Module	FX3
Performance	Up to 3050 Mbit/s
Ease of use	Turn on, start windows application, select destination, press button
Limitations	≈16 kB (one DMA buffer) get dropped randomly (after 1 Gbit on average, depends on host system)

Table 6.1.: Summary and comparison table for the different modules, with modules fulfilling the same or similar tasks shown next to each other.

input voltage swing is too small ($< 100\,\mathrm{mV}$) or the input common-mode voltage is in the range of 1 V to 1.5 V.

For the serial-to-parallel conversion the decision is more complicated. If the input speed is below 1330 Mbit/s, then the Shift Register PCB should be preferred unless the input frequency is in one of the unusable bands (as shown in fig. 4.3 on page 40). This is because the Shift Register PCB is easier to use and does not suffer from the repeated 32-bit words issue.
Above 1670 Mbit/s, the Demux PCB should be preferred, as it doesn't lose bits like the Shift Register PCB does at those speeds.
Between 1330 Mbit/s and 1670 Mbit/s there is only the Shift Register PCB in its semi-functional operating mode, and only in the range of 1480 Mbit/s to 1600 Mbit/s.

6.3. Outlook

While the interface is functional, its usability is compromised by data integrity issues. Further optimizations or alternative design choices are needed. For future work, I recommend pursuing an FPGA-based solution with the lessons learned in this thesis. The FX3 firmware could still be used and the Comparator and Level Shifter PCBs could be useful in FPGA-based designs as well. An FPGA-based solution could get rid of the data integrity and usability issues of the implemented serial-to-parallel conversion stage while increasing flexibility at the same time. This comes at the cost of yearly license fees for FPGA design software, which can be in the range of thousands of Euros or more per year.

If improving the implemented designs is preferable, several suggestions are listed in section 5.3 on page 53.

Appendix A.

Test Setups

The following measurement setups describe the setups used to independently verify the function and measure the performance of each part of the USB interface. The whole interface was also measured but is not pictured; it looks nearly identical to fig. 3.1 on page 8, just with the SPAD & Gater Circuit on the input replaced with a bit pattern generator (to rule out the SPAD as a source of errors).

The measurement setups for Demux and Shift Register PCBs (fig. A.1) as well as Comparator and Level Shifter PCBs (fig. A.2) are quite similar; both use a Centellax TG1C1-A clock synthesizer and a Sympuls Aachen BMG 12GIG bit pattern generator on the inputs. They differ in the outputs. Because the Demux and Shift Register PCBs have a parallel output bus, their outputs are measured with an Agilent 16823A logic analyzer. The Comparator and Level Shifter PCBs retain their high-speed differential serial signals and thus these can be fed to a Sympuls Aachen SBF 10GIG bit pattern receiver.

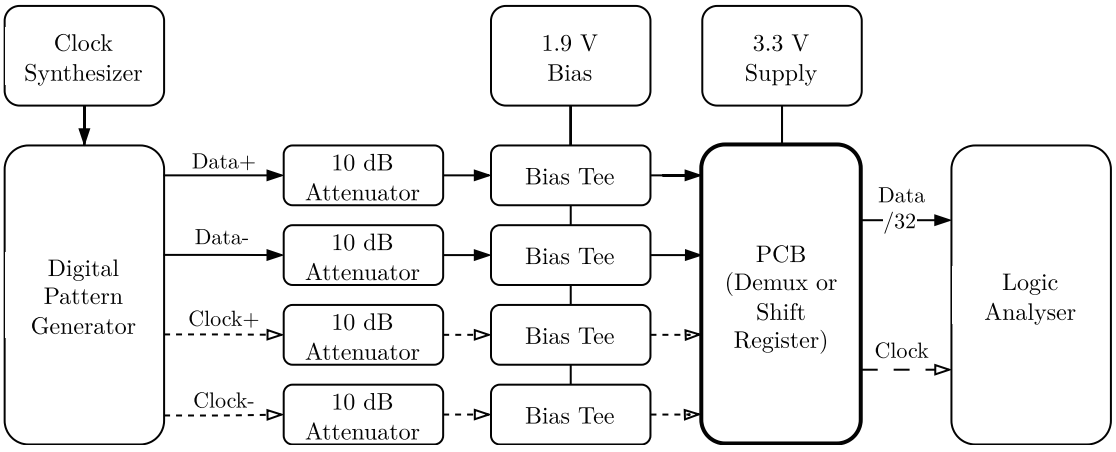


Figure A.1.: Measurement setup for independently measuring the Shift Register and Demux PCBs.

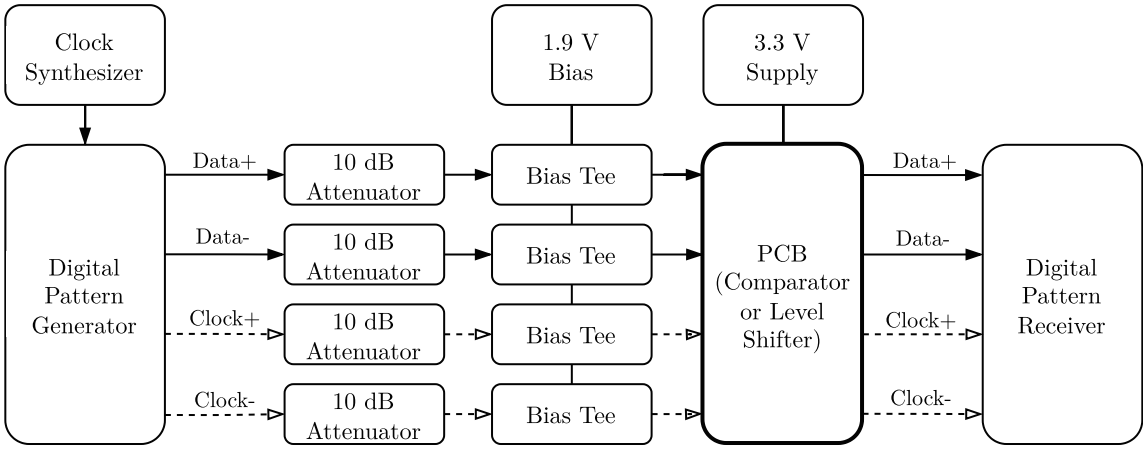


Figure A.2.: Measurement setup for independently measuring the Comparator and Level Shifter PCBs.

The measurement setup for the FX3 kit is shown in fig. A.3. The device on the left is an Agilent 16823A logic analyzer with a built-in pattern generator. Unfortunately, due to issues with signal integrity between the pattern generator outputs and the FX3 inputs, only speeds up to about 10 MHz bus speed or 320 Mbit/s could be tested independently. For higher speeds, at least the serial-to-parallel-conversion module had to be used together with the FX3.

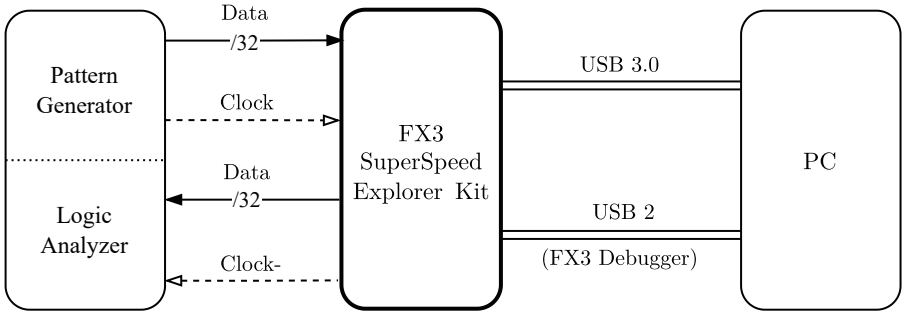


Figure A.3.: Measurement setup for independently measuring the FX3 SuperSpeed Explorer Kit.

Device	PC 1 ('lab')	PC 2 ('office')
Operating System	Microsoft Windows 10 Education	Microsoft Windows 10 Pro
OS Version	10.0.19045 Build 19045	10.0.19045 Build 19045
CPU	Intel Core i7-4770	AMD Ryzen 5 7600
Frequency	3400 MHz	3800 MHz
Core count	4	6
L1 Cache (per Core)	64 KB	64 KB
L2 Cache (per Core)	256 KB	1 MB
L3 Cache (shared)	8 MB	32 MB
RAM	32 GB DDR3-1600	32 GB DDR5-4800
BIOS version	Hewlett-Packard L01 v02.78, 20.02.2020	American Megatrends International, LLC. 1.28, 28.07.2023
Mainboard	Hewlett-Packard 18E4	ASRock B650M PG Lightning
Chipset	Intel Q87	AMD B650
USB 3 ports (directly connected to CPU)	0	4 ∗ SuperSpeed USB 10 Gbit/s
USB 3 ports (connected via chipset to CPU)	6 ∗ SuperSpeed 5 Gbit/s	5 ∗ SuperSpeed USB 10 Gbit/s

Table A.1.: Specifications for two PCs used for testing the USB interface.

Two PCs were used to evaluate the FX3: The 'lab' PC was used for most of the tests and showed the behaviour with missing DMA buffers, as described in section 5.2.5 on page 52. The 'office' PC was used during initial firmware development and to verify basic functionality at low transfer speeds (significantly below 500 Mbit/s). Because there was no access to a high-speed bit pattern generator near the 'office' PC, testing at higher transfer speeds, and especially with PRBS sequences, was not possible. Therefore, it could not be tested if the issue of missing DMA buffers discussed in section 5.2.5 on page 52 occurred less than with the 'lab' PC. Thus, it can only be stated that the FX3 also 'works' on the second ('office' PC) but no comparison is possible. Specifications for both devices are shown in A.1.

Appendix B.

PCB stackup considerations

Ott defines six design objectives for multilayer boards:

1. A signal layer should *always* be adjacent to a plane.
2. Signal layers should be tightly coupled (close) to their adjacent planes.
3. Power and ground planes should be closely coupled together.
4. High-speed signals should be routed on buried layers located between planes. The planes can then act as shields and contain the radiation from the high-speed traces.
5. Multiple-ground planes are very advantageous, because they will lower the ground (reference plane) impedance of the board and reduce the common-mode radiation.
6. When critical signals are routed on more than one layer, they should be confined to two layers adjacent to the same plane.

[33, p. 637]

The four-layer stackup used in this thesis (see fig. 3.9 on page 21) can only satisfy conditions #1 and #2. It could be argued that #5 is partly met as well in the areas where ground islands were used on the power layer.
A non-conventional approach would be to make the top and bottom planes both ground planes and use the two inner layers for signals as well as routed power. This would satisfy conditions #1, #2 and #4. This comes at the restriction of having to (fully) route power on the signal layers and it makes troubleshooting and board reworking much harder, so it was ruled out.

Appendix C.

Additional Figures and Data

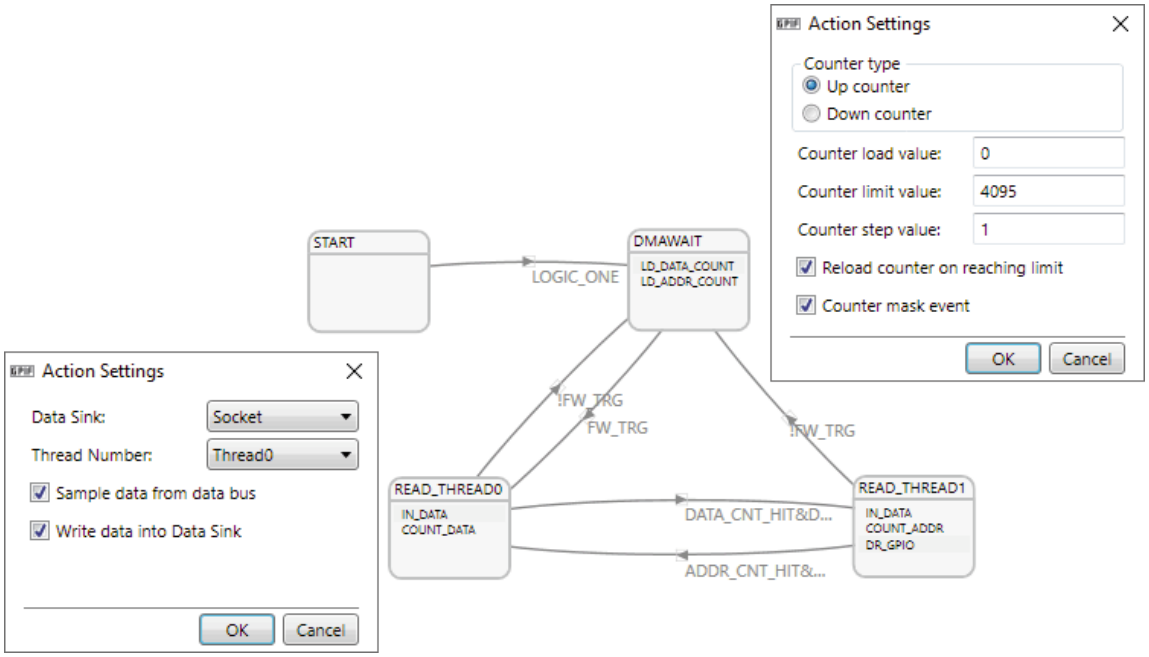


Figure C.1.: Screenshot of the GPIF II state machine within the GPIF II Designer application of the FX3 SDK. A more abstract representation (and explanation) is found in fig. 3.5 on page 13.

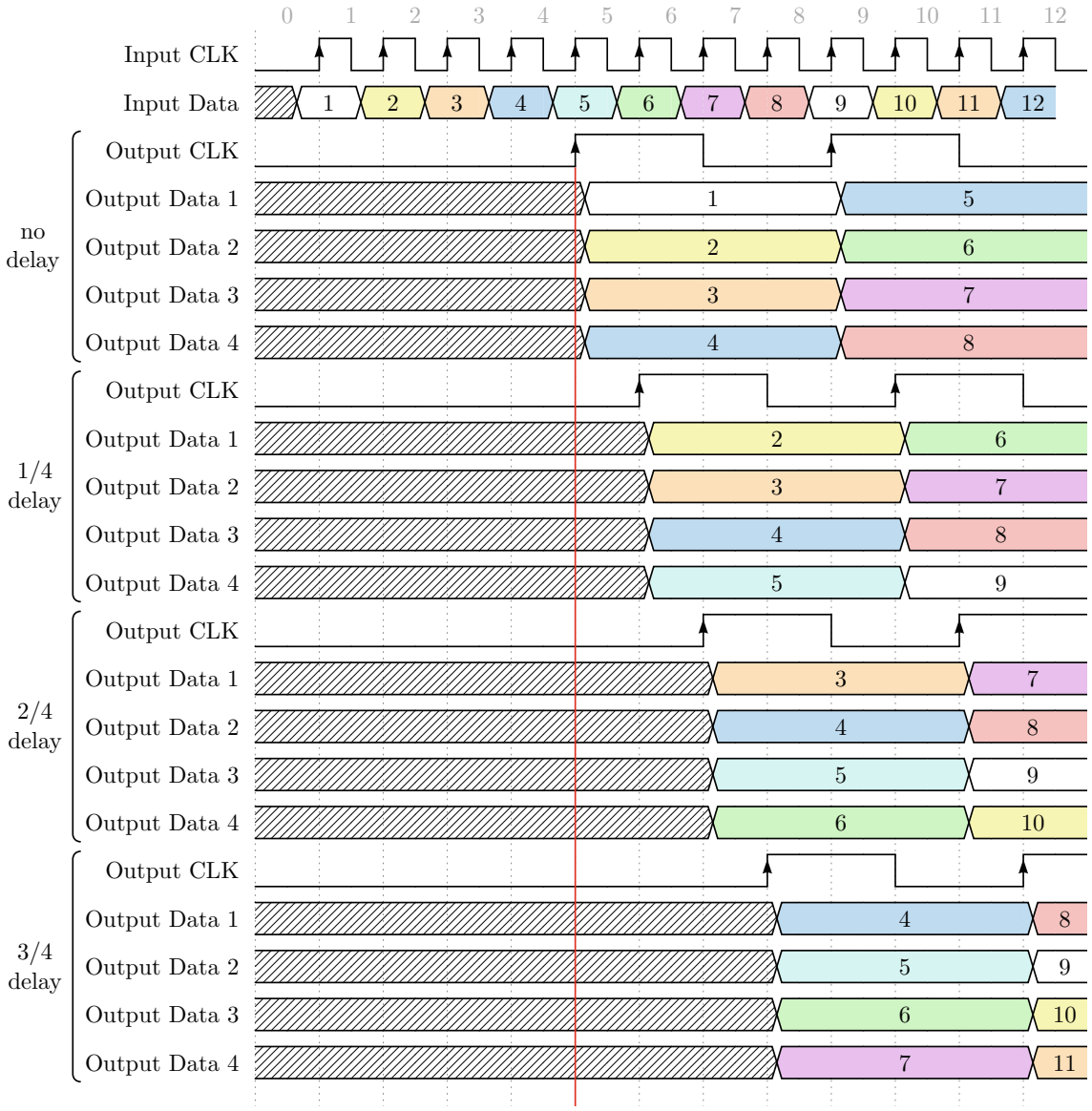
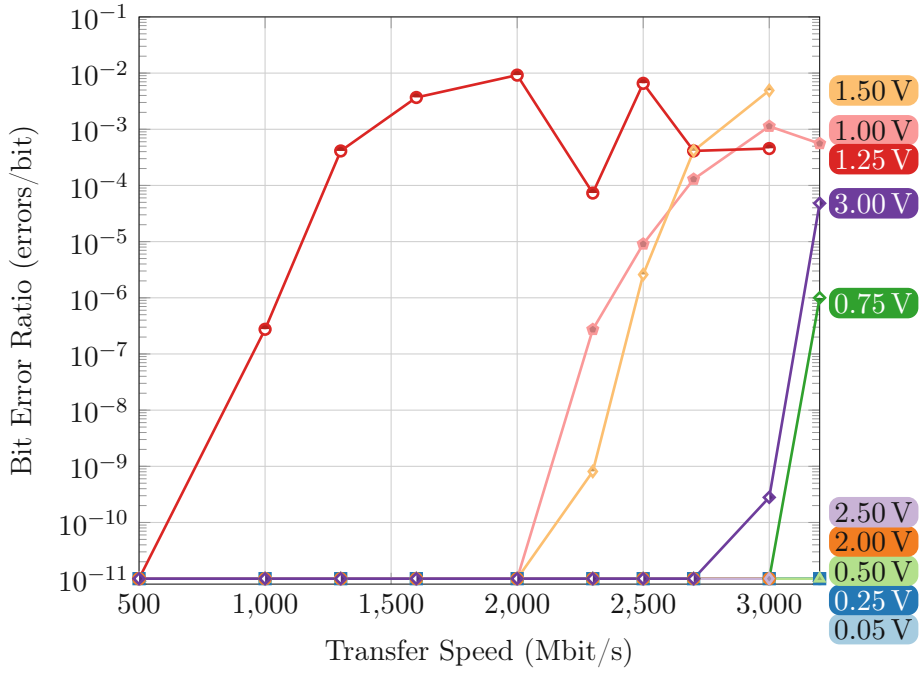
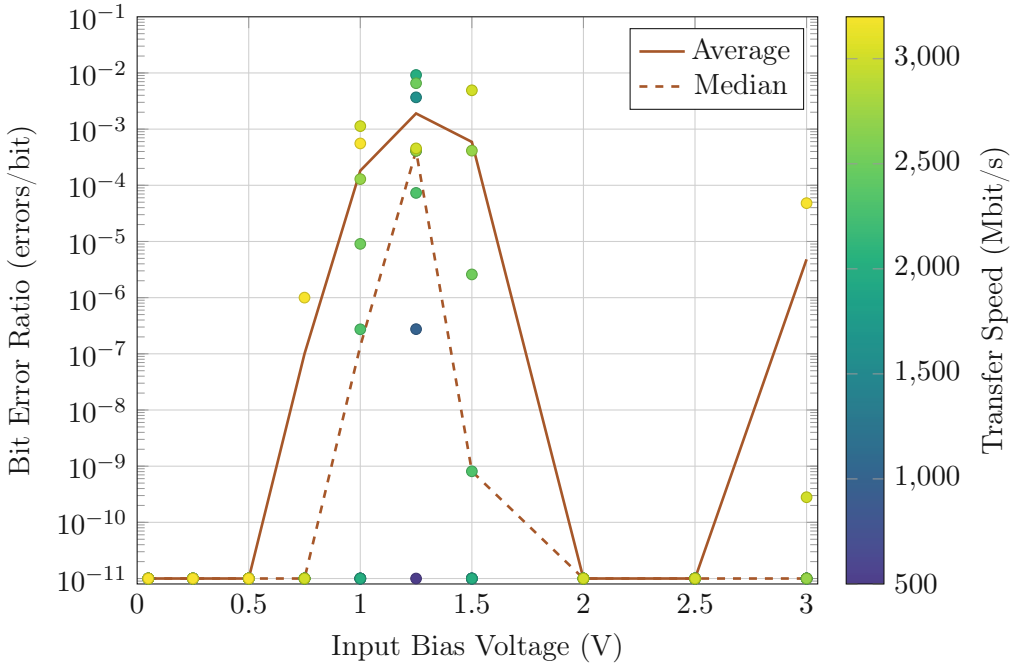


Figure C.2.: Four possible startup behaviours for a 1:4 deserializer. These differ in startup time, thereby changing which input bit is considered the first. Edges on the outputs shift accordingly. The first bit is output to the first output pin of the deserializer (Output Data 1), then the second (Output Data 2), and so on. Bit No. 5 is then output to the first output pin again. This is a more comprehensive variant of fig. 5.3 on page 51.



(a)



(b)

Figure C.3.: Transfer reliability for the Level Shifter PCB and a PRBS23 test sequence (instead of the more difficult PRBS31 sequence). Performance is strongly dependant on input bias voltage, with a stark increase in bit errors in the range of 1 V to 1.5 V. 1×10^{-11} errors/bit is the measurement limit, i.e. measurements like this meant no measured errors. Comparison with the PRBS31 variant in fig. 4.2 on page 38 shows an improvement at 1.00 V bias voltage, but mostly the same reliability otherwise.

Appendix D.

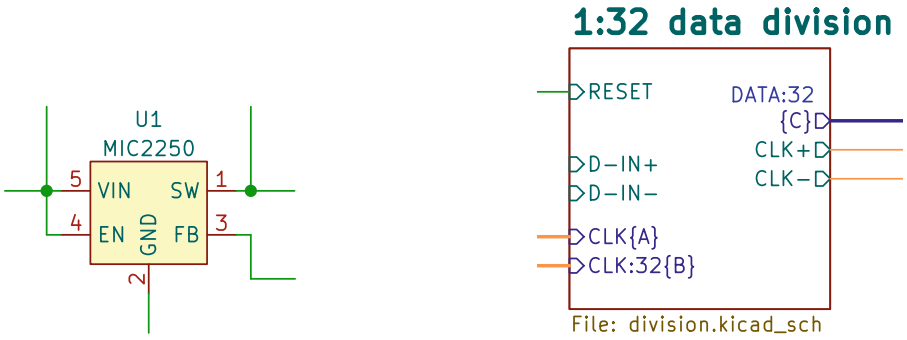
Schematics and Layouts

D.1. Comparator PCB Schematics

The following pages show the schematics as well as the layout of the Comparator PCB. Its structure and functionality is described in section 3.3.4 on page 29. The Shift Register PCB is only the MAX9376 level shifter IC (labelled U4 in the schematic), its associated bypass capacitors (C26, C27), the data SMA connectors (J1, J4, J8, J9) and the Power IN section of the schemativ (J7, C25, R6, R9, D1, D2). For this reason, schematics and layouts will be omitted as they can be inferred from the Comparator PCB schematics and layout.

Because the schematics are created with an EDA software (KiCad), they don't feature visible page numbers of the thesis but instead use their own internal numbering system. They are still counted as pages of the thesis however, so pages after the schematics feature the correct page number.

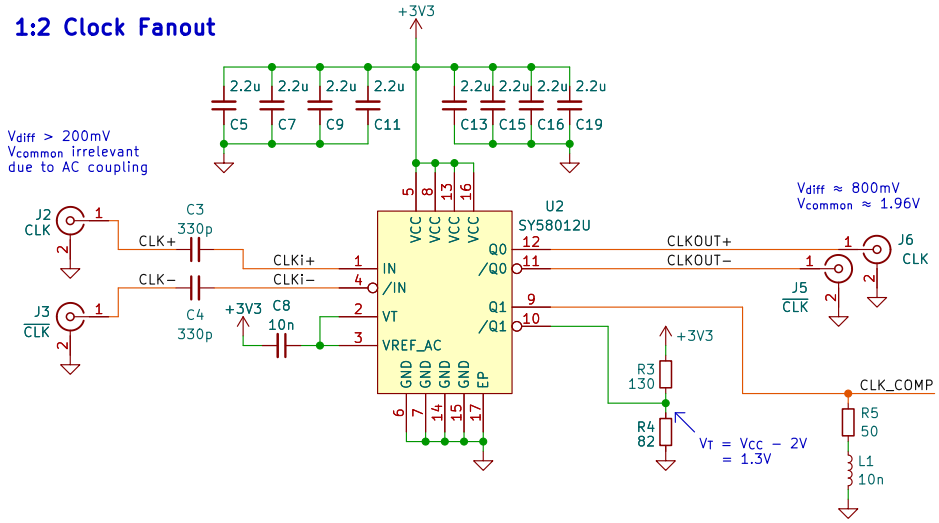
Note that the schematics follow a hierarchical design. Components have a yellow background colour, while hierarchical sheets have a white background colour (see fig. D.1). This is primarily a helpful tool in the used EDA software (KiCad) because it allows functional abstraction as well as reuse of repeating parts of the schematic.



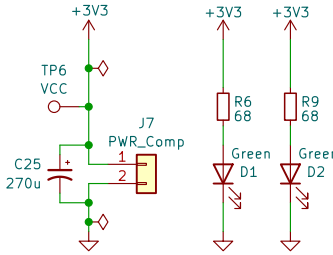
(a) Component (b) Hierarchical sheet

Figure D.1.: Examples for different schematic symbol types.

1:2 Clock Fanout

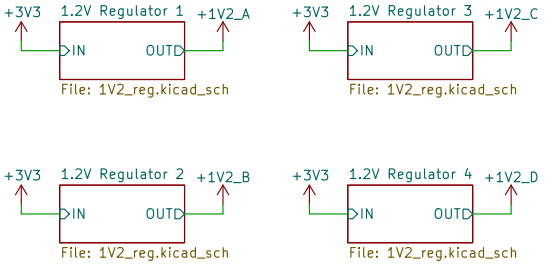


Power IN



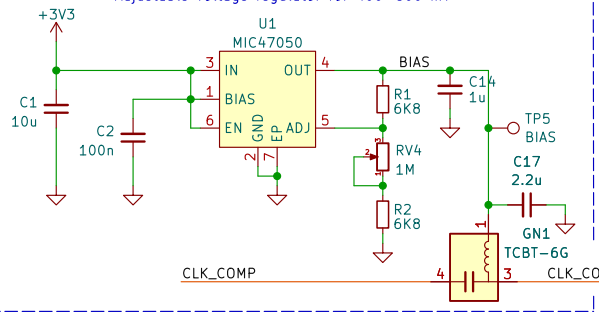
Voltage Regulators

Regulators for comparator



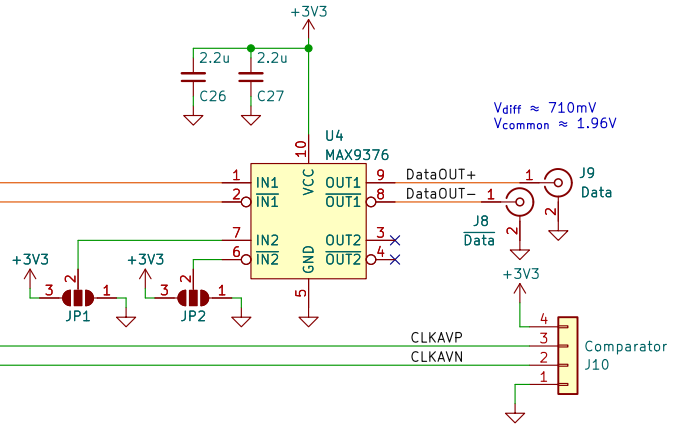
Bias Voltage

Adjustable voltage regulator for 400–800 mV



Custom Comparator

Level Shifter
1.2V NRZ –> LVPECL



Regular trace
Microstrip

H1 M3 H2 M3 H3 M3 H4 M3

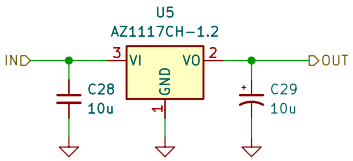
This circuit mainly consists of a custom high speed comparator and a subsequent level shifter that shifts the voltage to LVPECL levels. The input is a high–speed (up to 3Gbps) differential serial data stream as well as a clock.

Jakob Essbüchl | Vienna University of Technology

Sheet: /
File: Comparator–PCB.kicad_sch

Title: Custom Comparator

Size: A4	Date: 2023–08–07	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 1/5



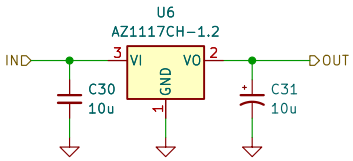
Linear voltage regulator to supply configuration voltage to the custom comparator chip

Jakob Essbüchl | Vienna University of Technology

Sheet: /1.2V Regulator 1/
File: 1V2_reg.kicad_sch

Title: 1.2 V Regulator

Size: User	Date: 2023-08-07	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 2/5



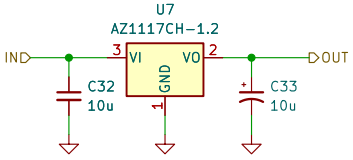
Linear voltage regulator to supply configuration voltage to the custom comparator chip

Jakob Essbüchl | Vienna University of Technology

Sheet: /1.2V Regulator 3/
File: 1V2_reg.kicad_sch

Title: 1.2 V Regulator

Size: User	Date: 2023-08-07	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 3/5



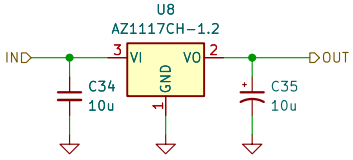
Linear voltage regulator to supply configuration voltage to the custom comparator chip

Jakob Essbüchl | Vienna University of Technology

Sheet: /1.2V Regulator 2/
File: 1V2_reg.kicad_sch

Title: 1.2 V Regulator

Size: User	Date: 2023-08-07	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 4/5



Linear voltage regulator to supply configuration voltage to the custom comparator chip

Jakob Essbüchl | Vienna University of Technology

Sheet: /1.2V Regulator 4/
File: 1V2_reg.kicad_sch

Title: 1.2 V Regulator

Size: User	Date: 2023-08-07	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 5/5

D.2. Comparator PCB Layout

All layout figures are to scale and viewed from the top.

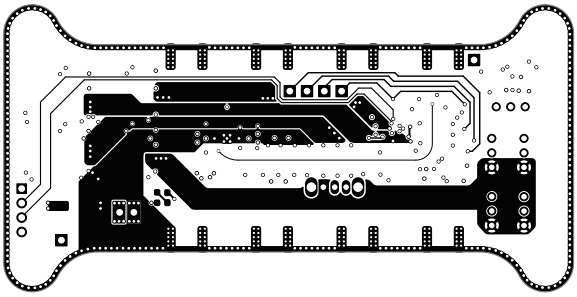


Figure D.2.: Top side copper of the Comparator PCB. Used for signals as well as voltage rails.

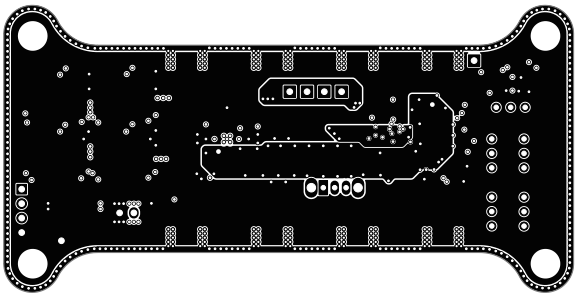


Figure D.3.: Top inner layer (In1) copper of the Comparator PCB. Used for voltage rails as well as a ground island.

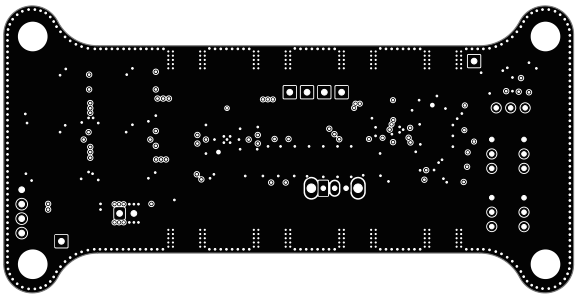


Figure D.4.: Bottom inner layer (In2) copper of the Comparator PCB. Used as a ground plane.

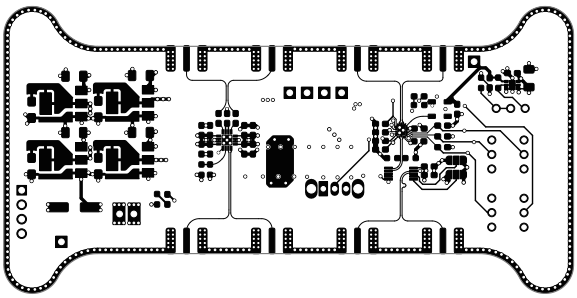


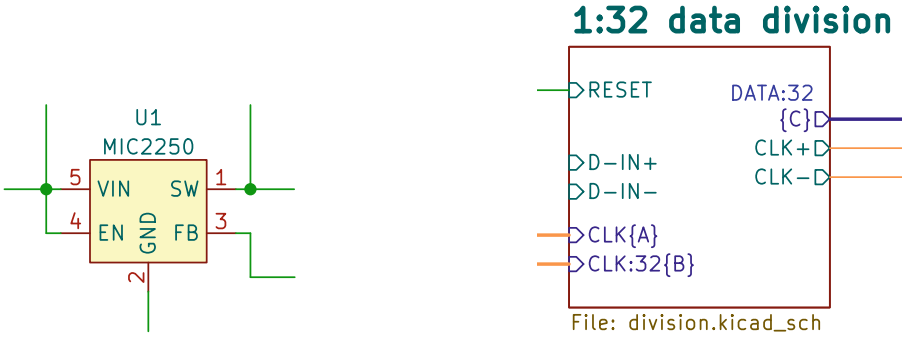
Figure D.5.: Bottom side copper of the Comparator PCB. Used for signals.

D.3. Shift Register PCB Schematics

The following pages show the schematics as well as the layout of the Shift Register PCB. Its structure and functionality is described in section 3.3.2 on page 22.

Because the schematics are created with an EDA software (KiCad), they don't feature visible page numbers of the thesis but instead use their own internal numbering system. They are still counted as pages of the thesis however, so pages after the schematics feature the correct page number.

Note that the schematics follow a hierarchical design. Components have a yellow background colour, while hierarchical sheets have a white background colour (see fig. D.6). This is primarily a helpful tool in the used EDA software (KiCad) because it allows functional abstraction as well as reuse of repeating parts of the schematic.



(a) Component (b) Hierarchical sheet

Figure D.6.: Examples for different schematic symbol types.

Regular trace
Microstrip

Data IN LVPECL levels

J2 Data
J6 Data

1:32 data division

RESET
RESET
D−IN+
D−IN−
CLK{A}
CLK:32{B}
DATA:32 {C}
CLK+
CLK−

File: division.kicad_sch

Level Shifting

DATA:32 {C}
CLK_PECL+
CLK_PECL−
DATA_TTL[0..31]
CLK_TTL

File: level_shifting.kicad_sch

FX3

DATA[0..31]
CLK

File: fx3.kicad_sch

Clock IN LVPECL levels

J1 CLK
J4 CLK

1:32 Clock division

RESET
RESET
CLK+
CLK−
SDAT
SCLK
nCS
CLK{A}
CLK:32{B}

File: clock_division.kicad_sch

Power IN

J13 VCC
J14 GND
C5 270u
J16 Supply
C6 270u
R12 68
R13 68
D5 Green
D6 Green

Two different ways of connecting power
l: 4mm Banana Plugs r: Screw terminals

LEDs on both top and bottom

Adjustable 5−12V boost regulator for cooling fan

L1 10u
U1 MIC2250
D4 1N4148W
TP1 VFAN
R7 100K
RV1 30K
R8 11K
C2 10u
C3 10u

Regulator for ECL termination voltage VCC−2V

https://www.analog.com/en/design-notes/modified-ldo-regulator-sinks-pecltermination-current.html

C4 1u
R10 150K
R11 91K
$V_T = V_{CC} - 2V = 1.3V$
U2 MAX1735
Q2 BD438
HS1 Heatsink
C7 10u

USB−C connector for programming microcontroller

J5 USB
R1 5k11
R2 5k11
C1 100n
S1

STM32 Microcontroller

USB_D−
USB_D+
NRST
SWCLK
SWDIO
PWM_FAN2
PWM_FAN
FAN_RPM
SPI_MOSI
SPI_SCK
SPI_NSS
IO1
IO2
I2C_SDA
I2C_SCL
UART_RX
UART_TX
LED_RDY

File: microcontroller.kicad_sch

Programming and debug port
J8 SWD

J10 IO
J11 I2C
J12 UART
R5
R6

J7 Fan
D1 1N4148W
Q1 DMN3731U
JP1 Fan
J9 Fan

2 variants of cooling fans:
4 pin and 2 pin fans

R3 68
R4 68
D2 Green
D3 Green

LEDs on both top and bottom

Supplementary I/Os

Power OUT

JST−XH connector to supply power to comparator PCB

J3 Power

Inversion for active−low resets

J15 Reset
SW1 RESET
R9 10K
U3 SN74LVC1G04

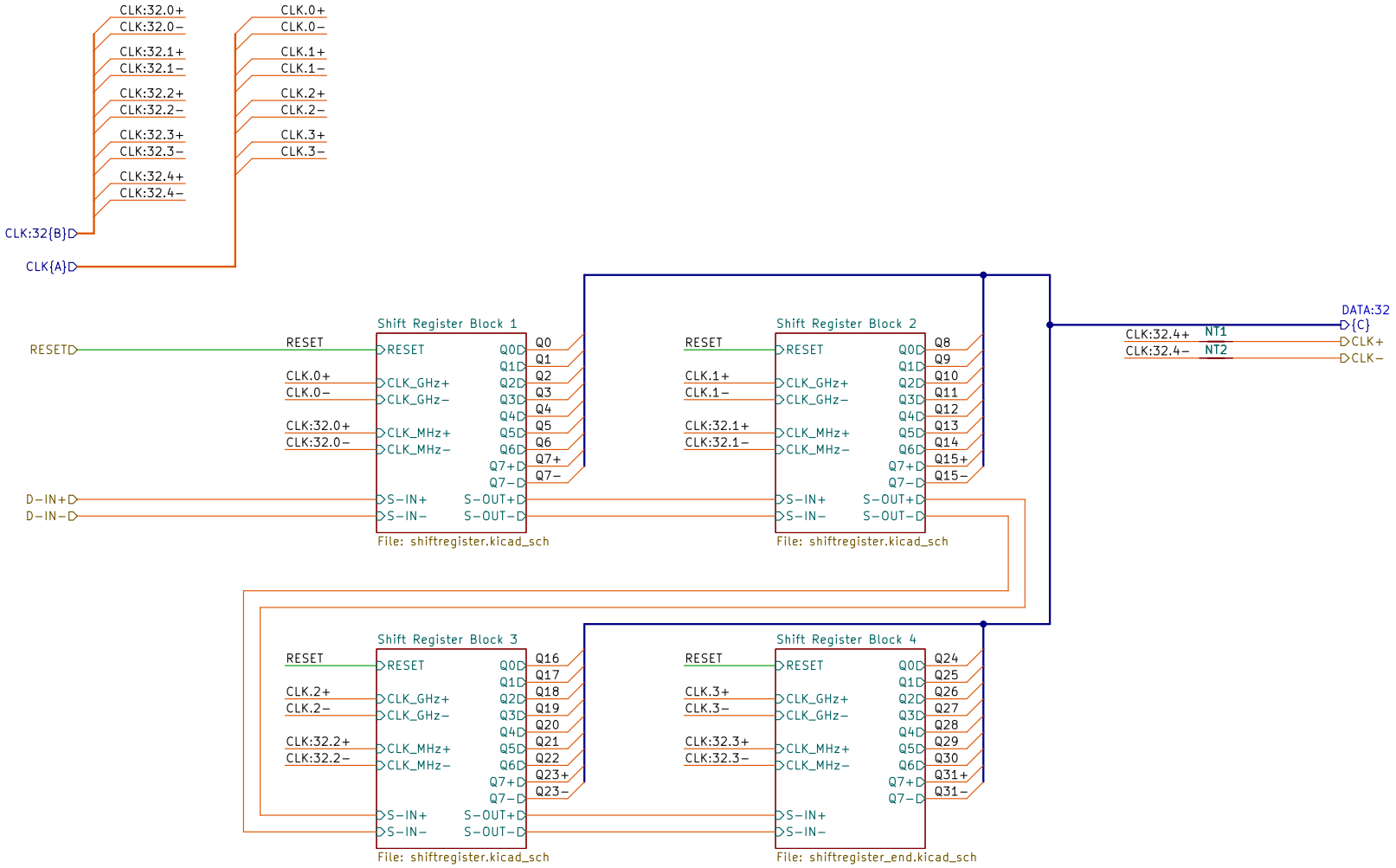
This circuit consists of a daisy−chained shift register array that converts a differential high−speed serial data stream into 32 parallel data lines. The input high−speed clock is also divided by 32. Parallelized data + divided clock are then sent to the FX3 USB interface. A microcontroller handles configuration of the programmable clock divider.

Jakob Essbüchl | Vienna University of Technology

Sheet: /
File: Shift−Register−Shield.kicad_sch

Title: Shift Register PCB

Size: A4	Date: 2023−07−13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 1/27



Regular trace
Microstrip

This circuit takes a high−frequency (<3 Gbit/s) data signal and divides it to 32 parallel lines.

Inputs and outputs are using LVPECL signal levels.

Jakob Essbüchl | Vienna University of Technology

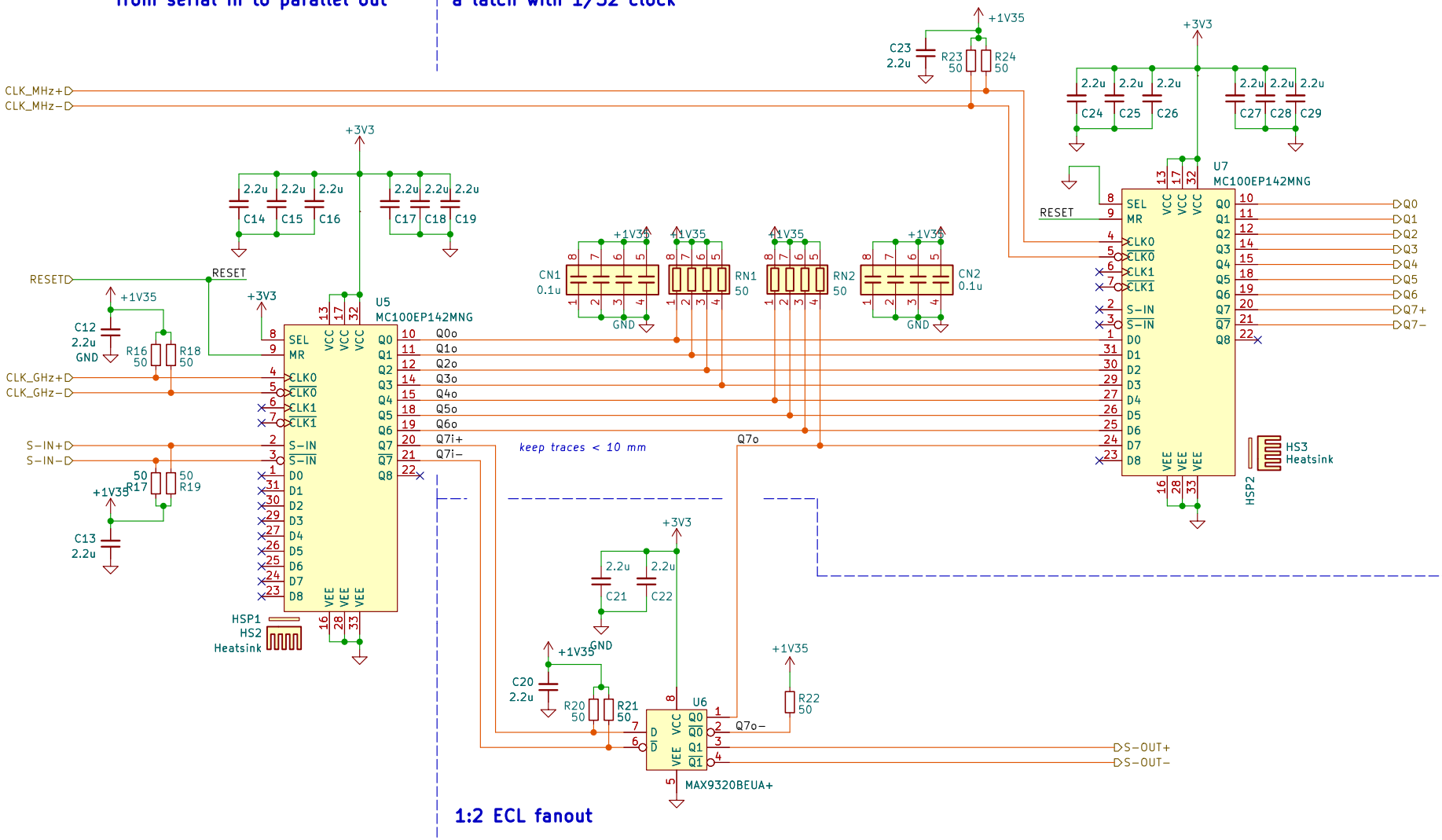
Sheet: /1:32 data division/
File: division.kicad_sch

Title: Data Division Serial to Parallel

Size: A4	Date: 2023−07−13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 2/27

9-bit Shift register for shifting from serial in to parallel out

9-bit Shift register used as a latch with 1/32 clock



1:2 ECL fanout

This circuit contains two 8-bit shift registers. The first one acts as a shifter but has a non-latching output, i.e. the 8 parallel outputs are shifted by 1 bit each but are still at the input frequency.
The second shift register acts as a latch, being triggered by the 1/32 divided clock.

Jakob Essbüchl | Vienna University of Technology

Sheet: /1:32 data division/Shift Register Block 1/
File: shiftregister.kicad_sch

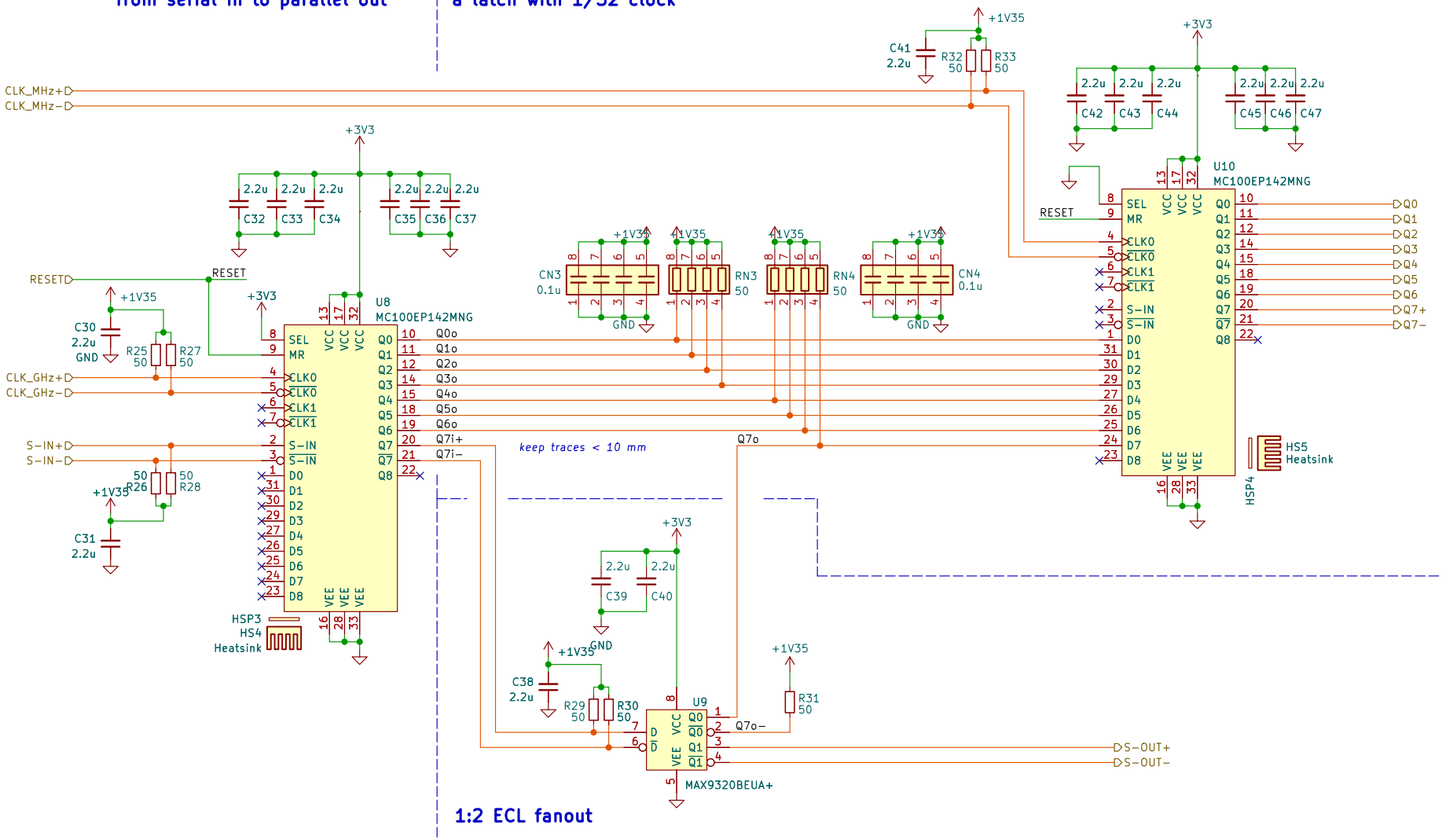
Title: Shift Register Block

Size: A4	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 3/27

Regular trace
Microstrip

9-bit Shift register for shifting from serial in to parallel out

9-bit Shift register used as a latch with 1/32 clock



1:2 ECL fanout

Regular trace
Microstrip

This circuit contains two 8-bit shift registers. The first one acts as a shifter but has a non-latching output, i.e. the 8 parallel outputs are shifted by 1 bit each but are still at the input frequency.
The second shift register acts as a latch, being triggered by the 1/32 divided clock.

Jakob Essbüchl | Vienna University of Technology

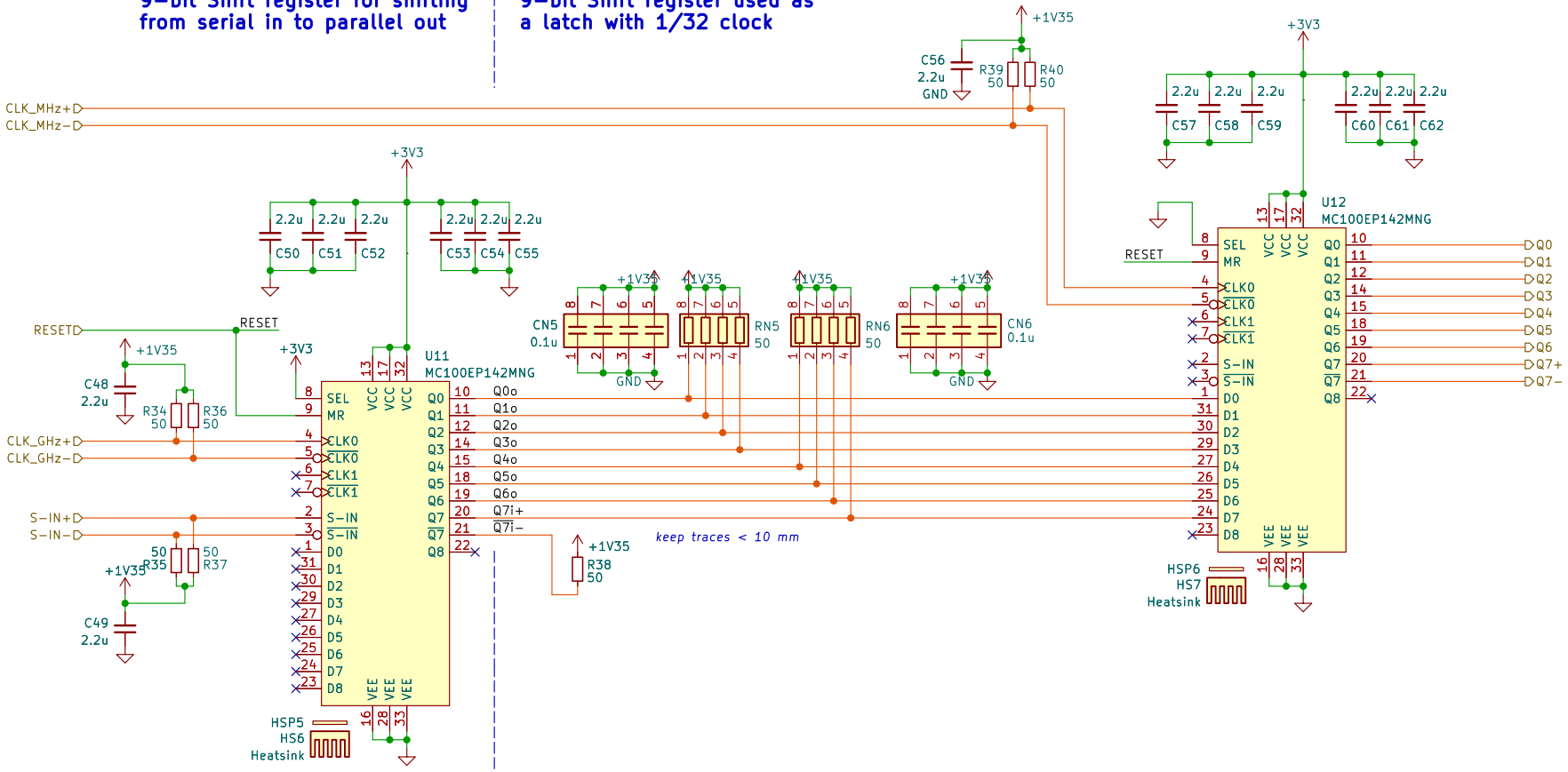
Sheet: /1:32 data division/Shift Register Block 3/
File: shiftregister.kicad_sch

Title: Shift Register Block

Size: A4	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 4/27

9-bit Shift register for shifting from serial in to parallel out

9-bit Shift register used as a latch with 1/32 clock



This circuit contains two 8-bit shift registers. The first one acts as a shifter but has a non-latching output, i.e. the 8 parallel outputs are shifted by 1 bit each but are still at the input frequency.
The second shift register acts as a latch, being triggered by the 1/32 divided clock.

Jakob Essbüchl | Vienna University of Technology

Sheet: /1:32 data division/Shift Register Block 4/
File: shiftregister_end.kicad_sch

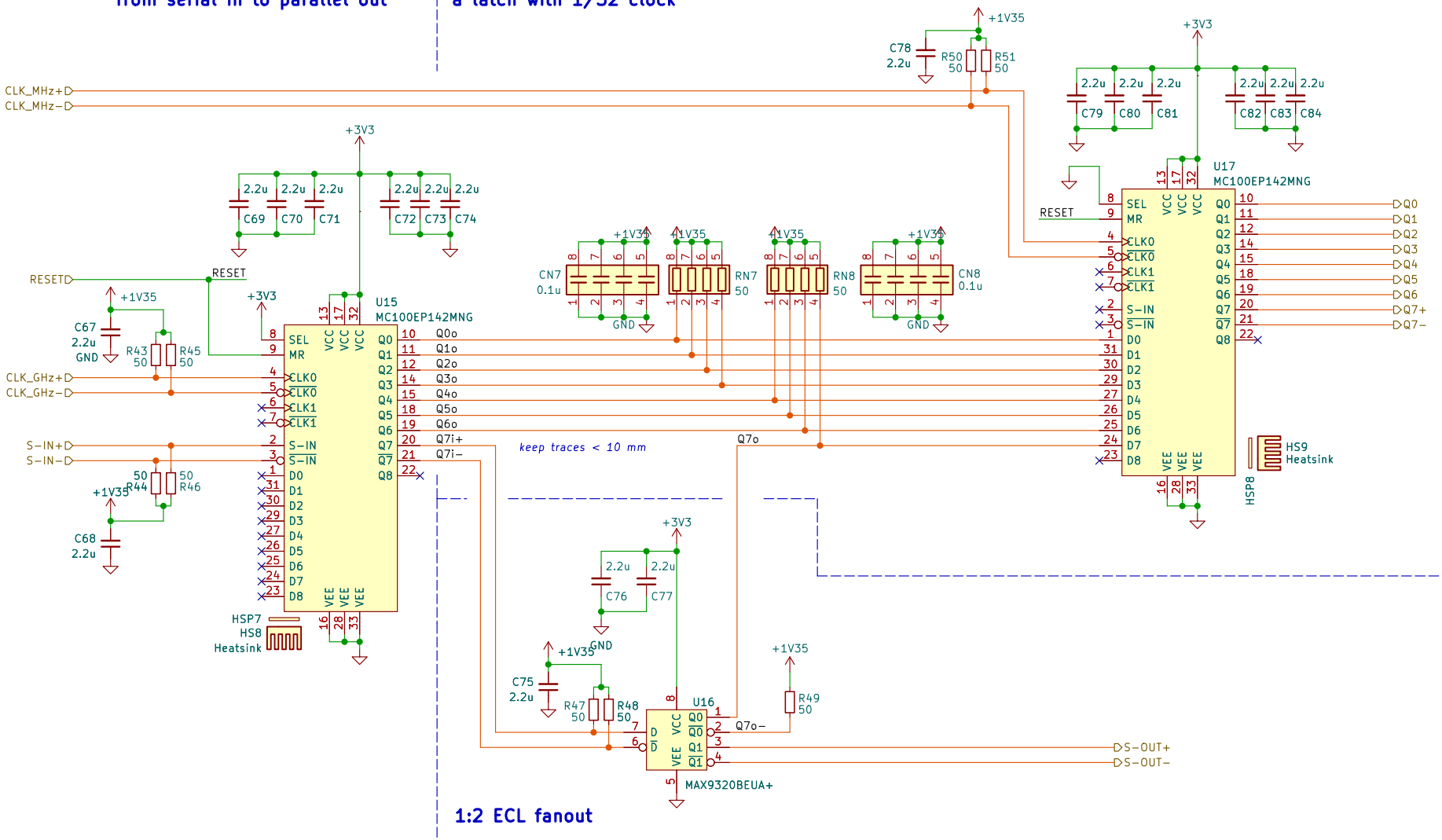
Title: Shift Register Block

Size: A4	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 5/27

Regular trace
Microstrip

9-bit Shift register for shifting from serial in to parallel out

9-bit Shift register used as a latch with 1/32 clock



1:2 ECL fanout

Regular trace
Microstrip

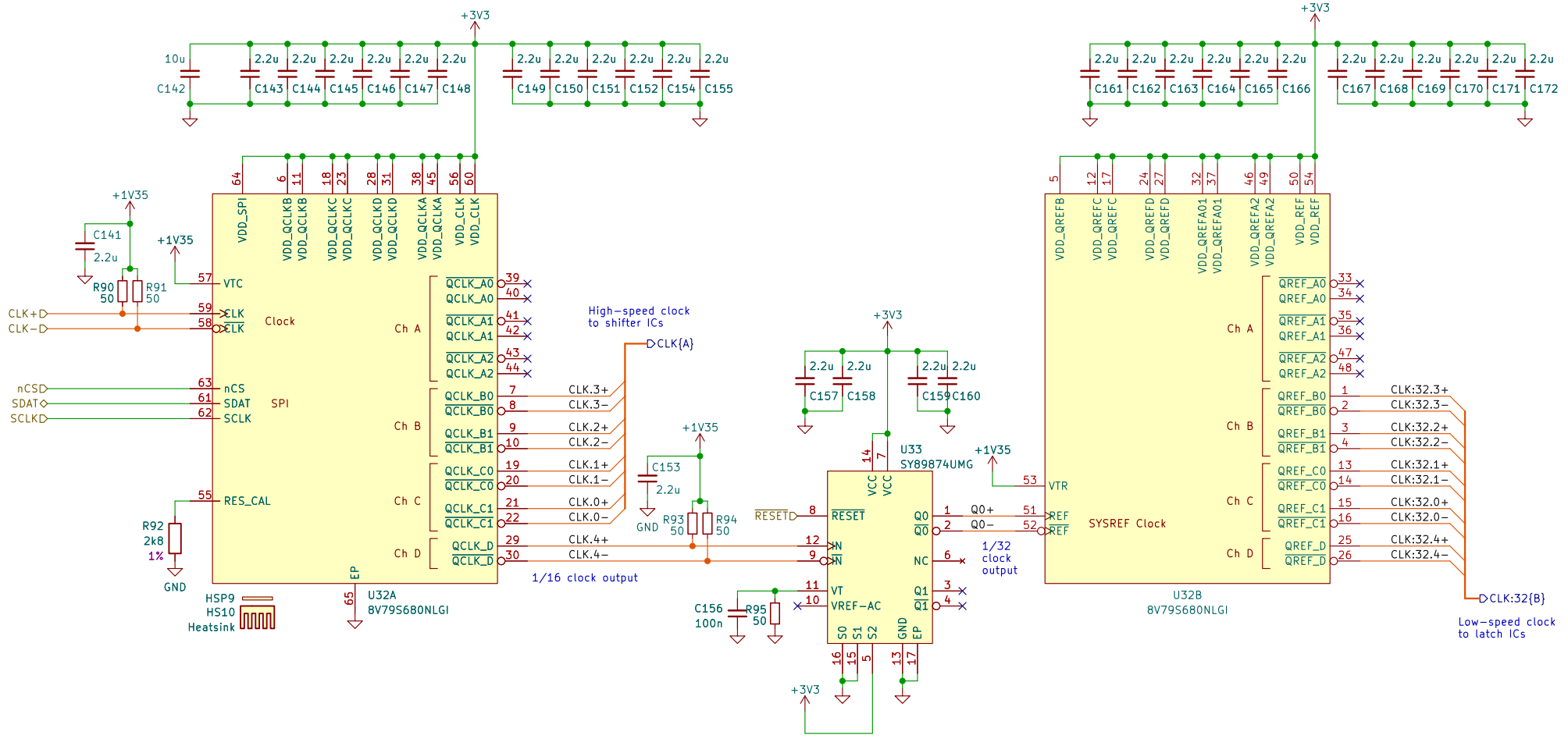
This circuit contains two 8-bit shift registers. The first one acts as a shifter but has a non-latching output, i.e. the 8 parallel outputs are shifted by 1 bit each but are still at the input frequency.
The second shift register acts as a latch, being triggered by the 1/32 divided clock.

Jakob Essbüchl | Vienna University of Technology

Sheet: /1:32 data division/Shift Register Block 2/
File: shiftregister.kicad_sch

Title: Shift Register Block

Size: A4	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 6/27



High−speed clock distribution

1/2 clock divider

Low−speed clock distribution

This circuit takes a high−frequency (<3 GHz) clock and divides it to 1:32.
It also acts as a fanout buffer as well as a configurable delay.

Inputs and outputs are using LVPECL signal levels.

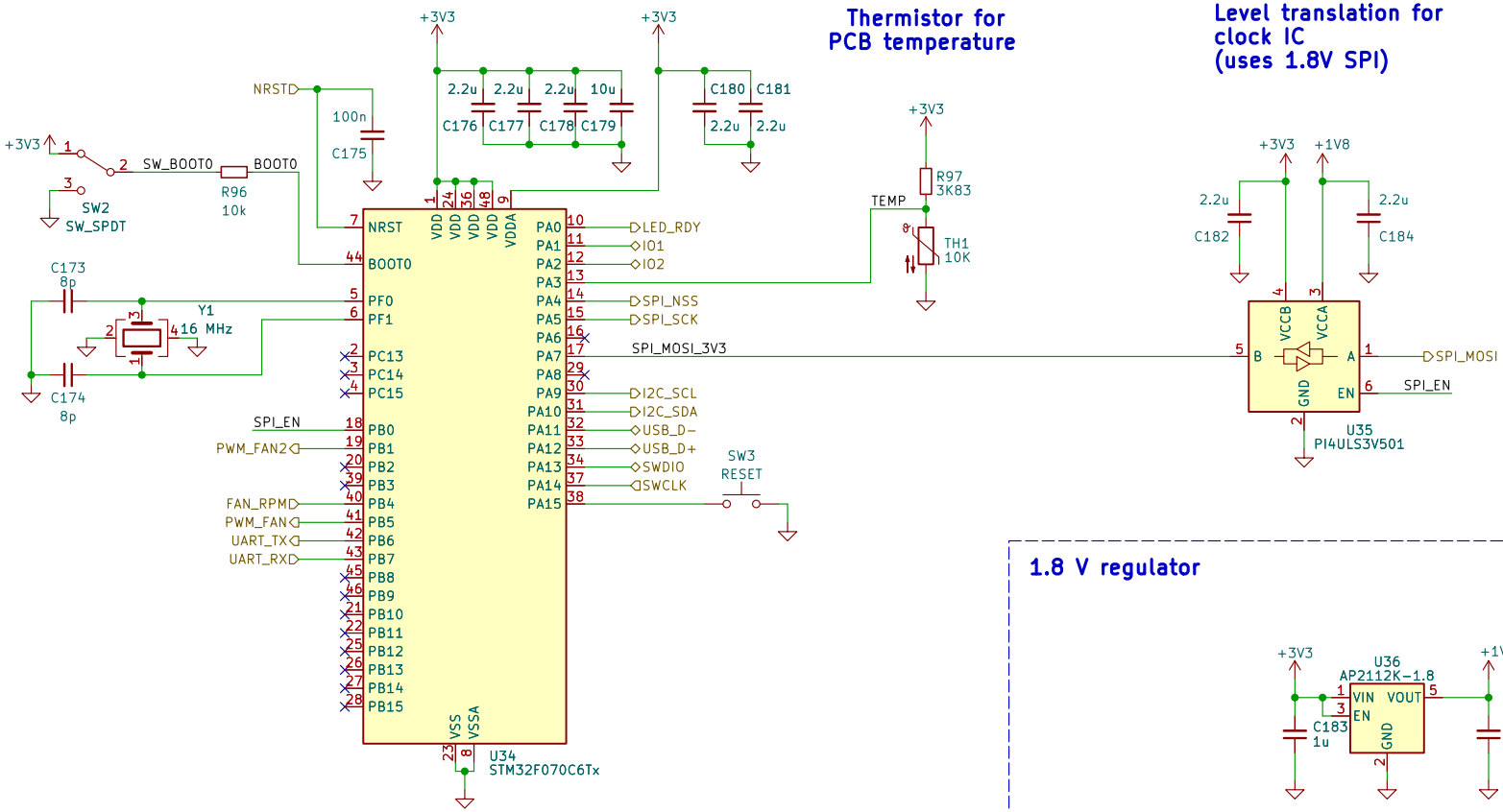
Jakob Essbüchl | Vienna University of Technology

Sheet: /1:32 Clock division/
File: clock_division.kicad_sch

Title: 1:32 Clock Division

Size: A4	Date: 2023−07−13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 7/27

Microcontroller to configure clock IC and control cooling fan



Thermistor for PCB temperature

Level translation for clock IC (uses 1.8V SPI)

1.8 V regulator

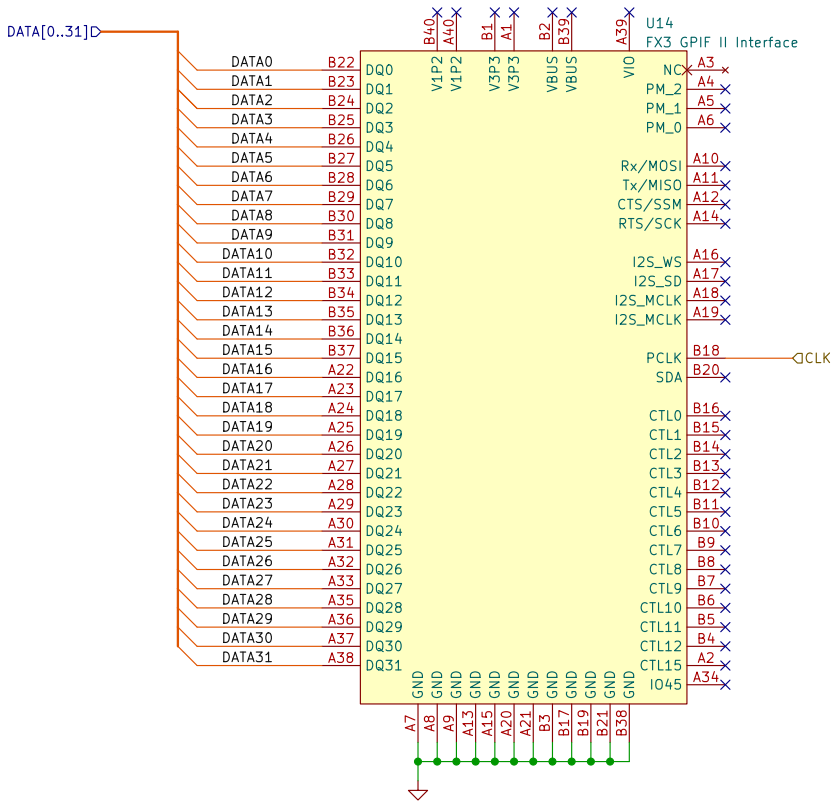
This circuit contains the STM32 microcontroller and some of the associacted components.
The uC configures and calibrates the clock IC via SPI upon startup. It can also control a cooling fan depending on board temperature.
Since the clock IC uses 1.8V SPI some level translation is needed.

Jakob Essbüchl | Vienna University of Technology

Sheet: /STM32 Microcontroller/
File: microcontroller.kicad_sch

Title: STM32 Microcontroller

Size: A4	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 8/27



Regular trace
Microstrip

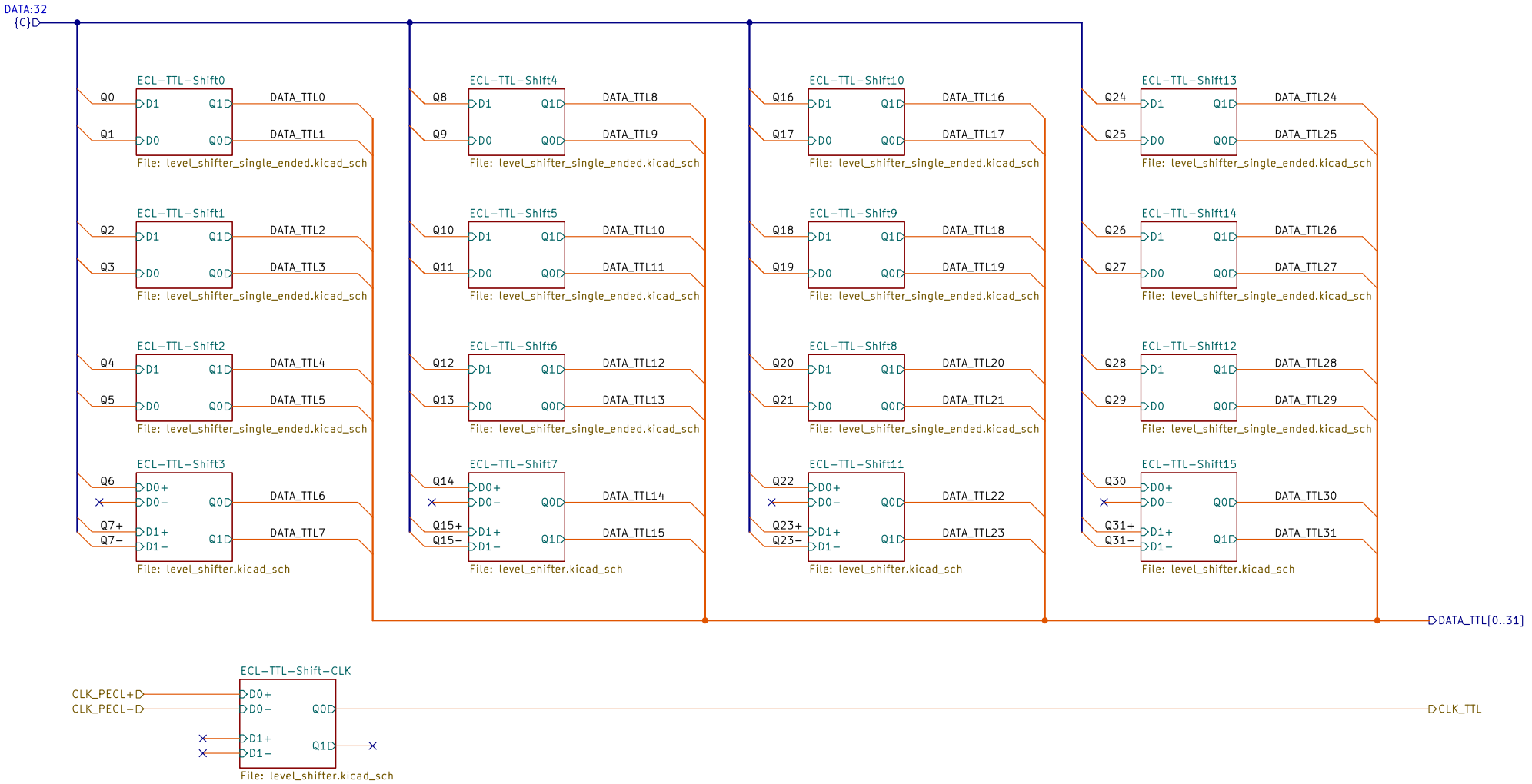
The FX3 facilitates the USB3 connection to the PC.
It connects to the PCB via it's GPIFII interface.

Jakob Essbüchl | Vienna University of Technology

Sheet: /FX3/
File: fx3.kicad_sch

Title: FX3 SuperSpeed Development Kit GPIFII connector

Size: A4	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 9/27



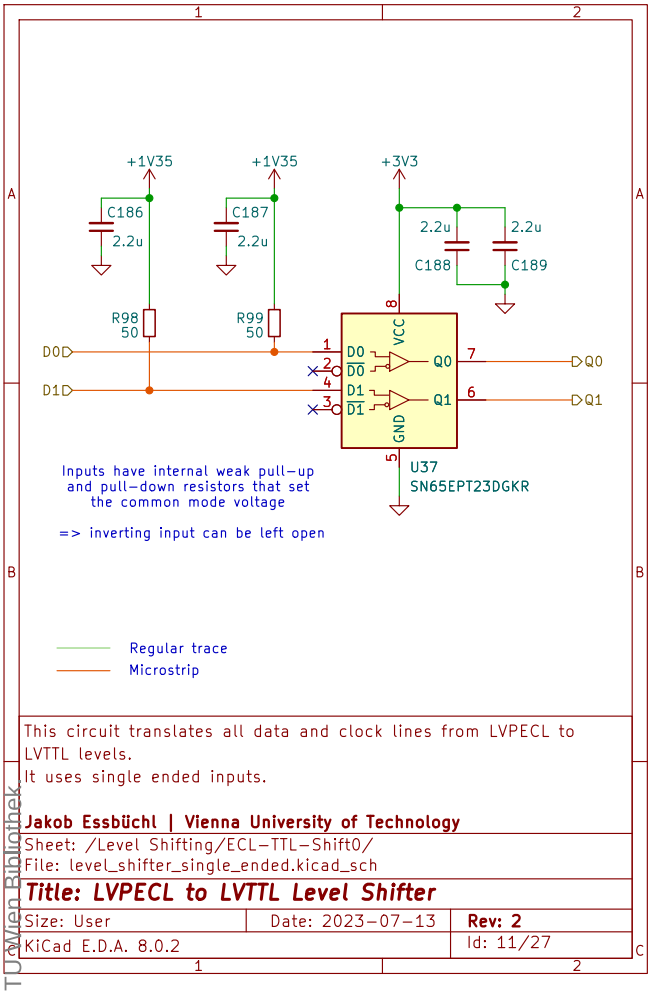
This circuit translates all data and clock lines from LVPECL to LVTTL levels.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/
File: level_shifting.kicad_sch

Title: Level Shifting Array

Size: A4	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 10/27



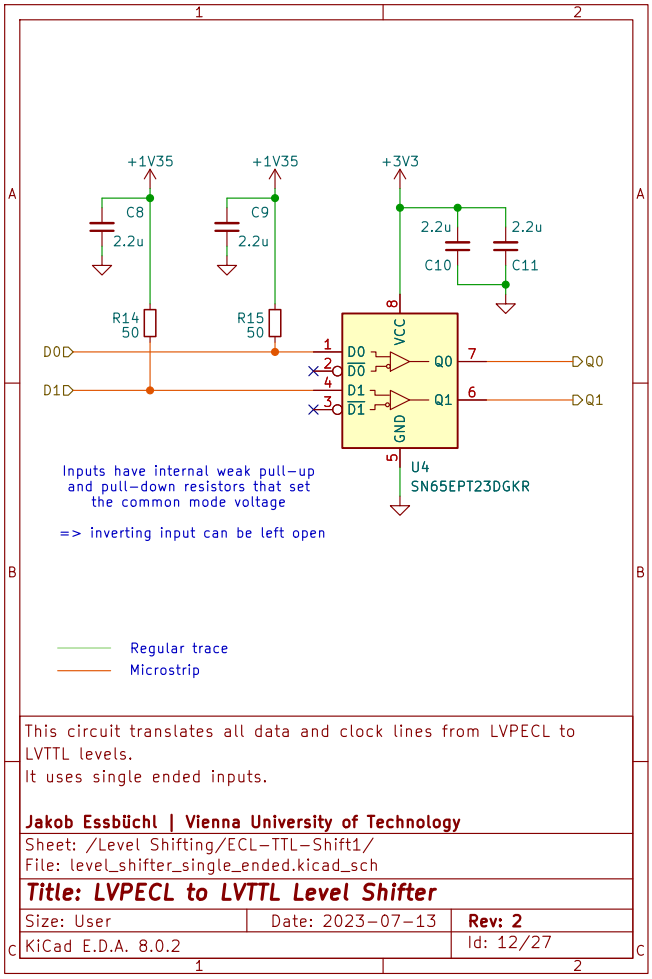
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift0/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 11/27



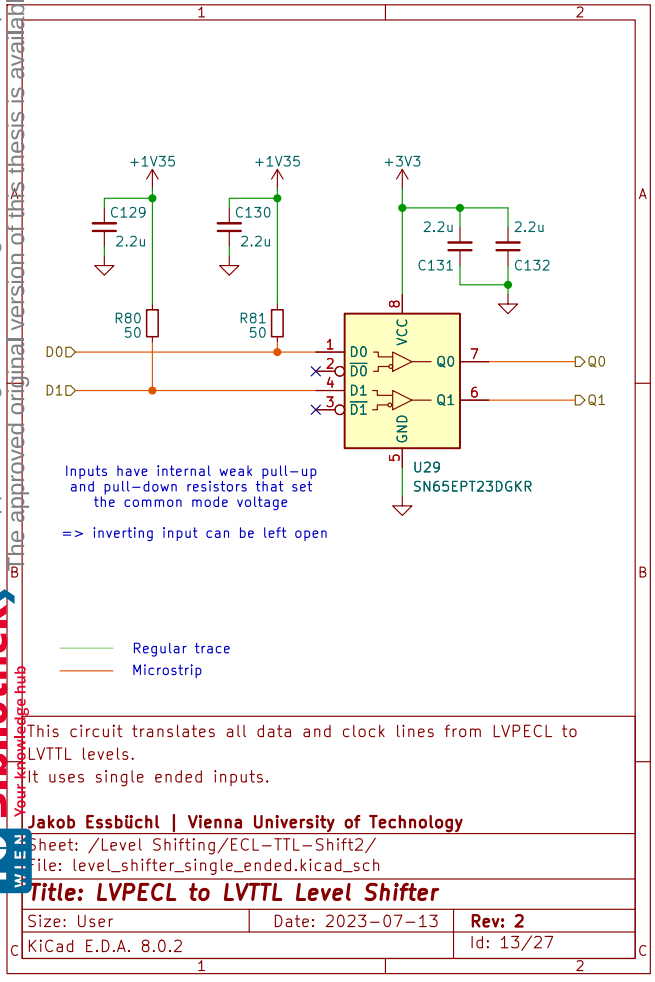
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift1/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 12/27



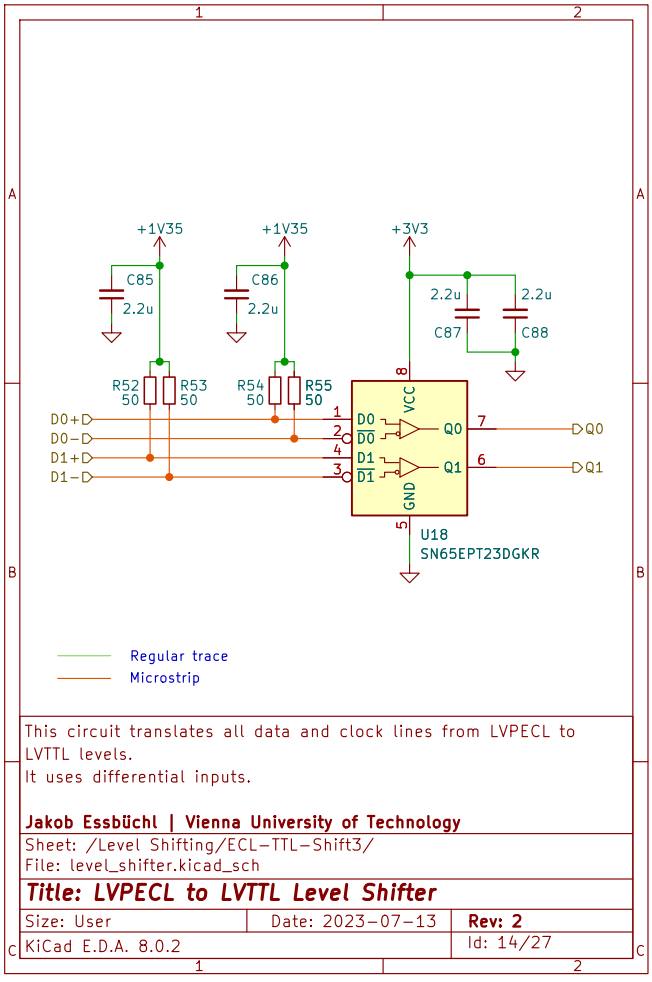
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift2/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 13/27



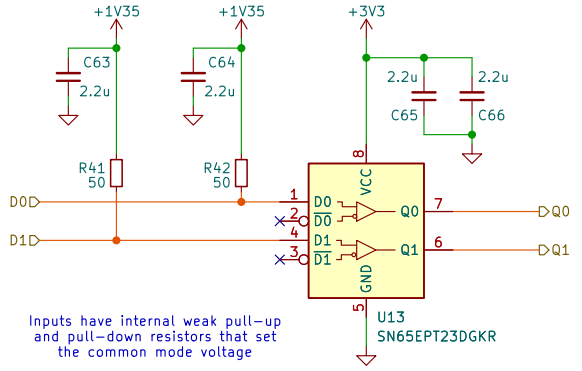
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses differential inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift3/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 14/27



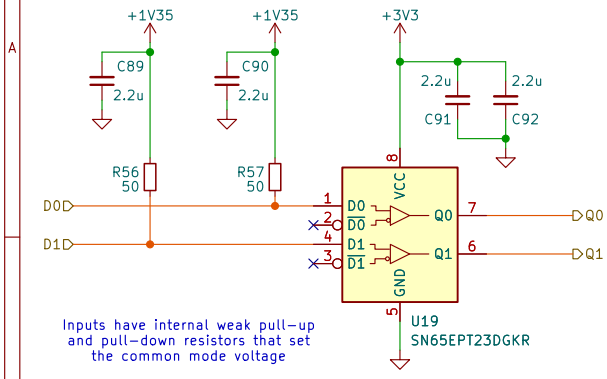
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift4/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 15/27



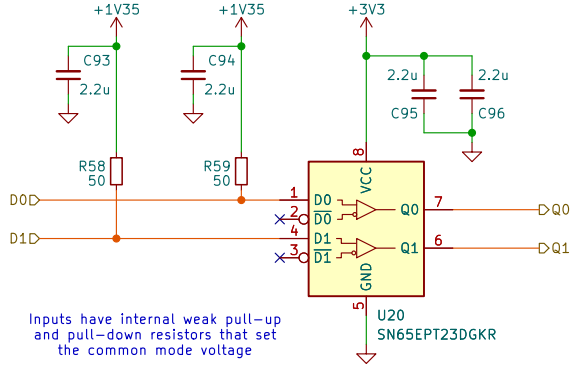
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift5/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 16/27



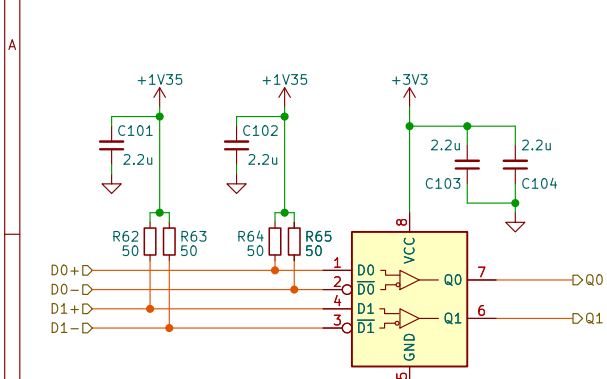
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift6/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 17/27



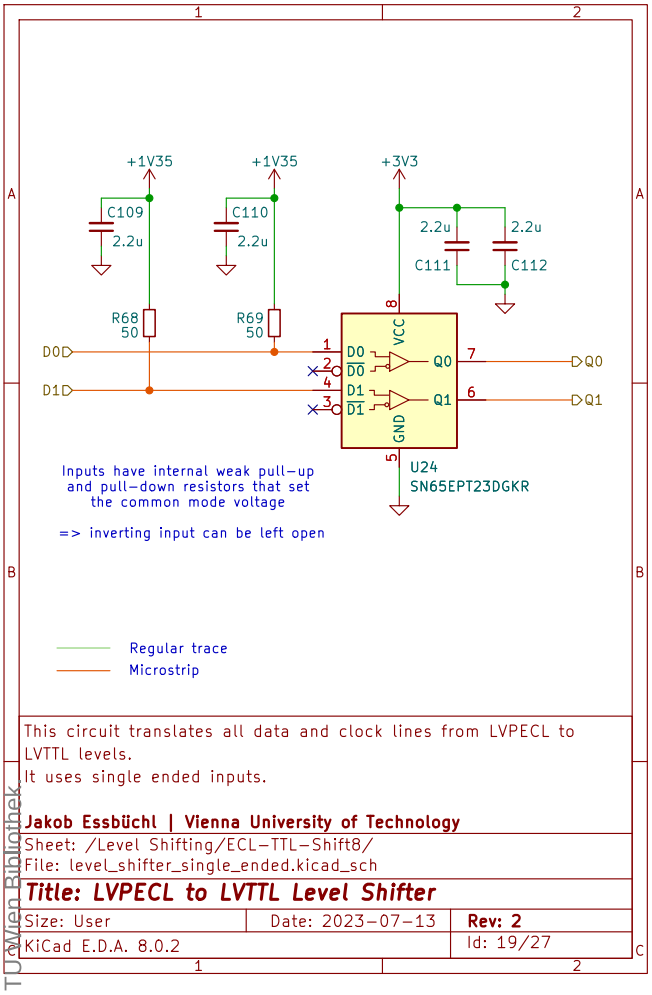
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses differential inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift7/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 18/27



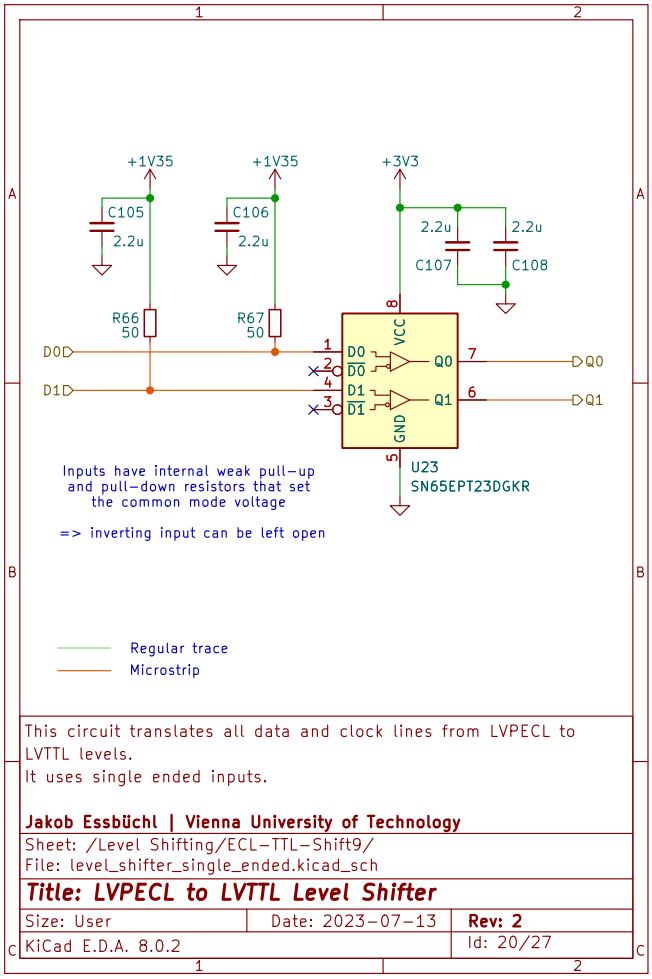
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift8/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 19/27



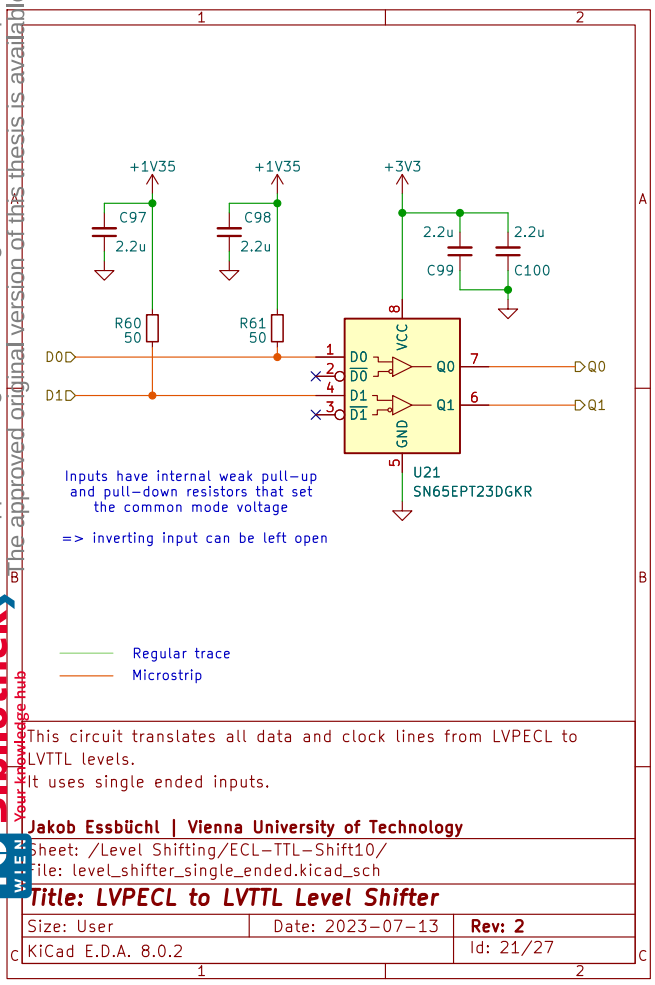
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift9/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 20/27



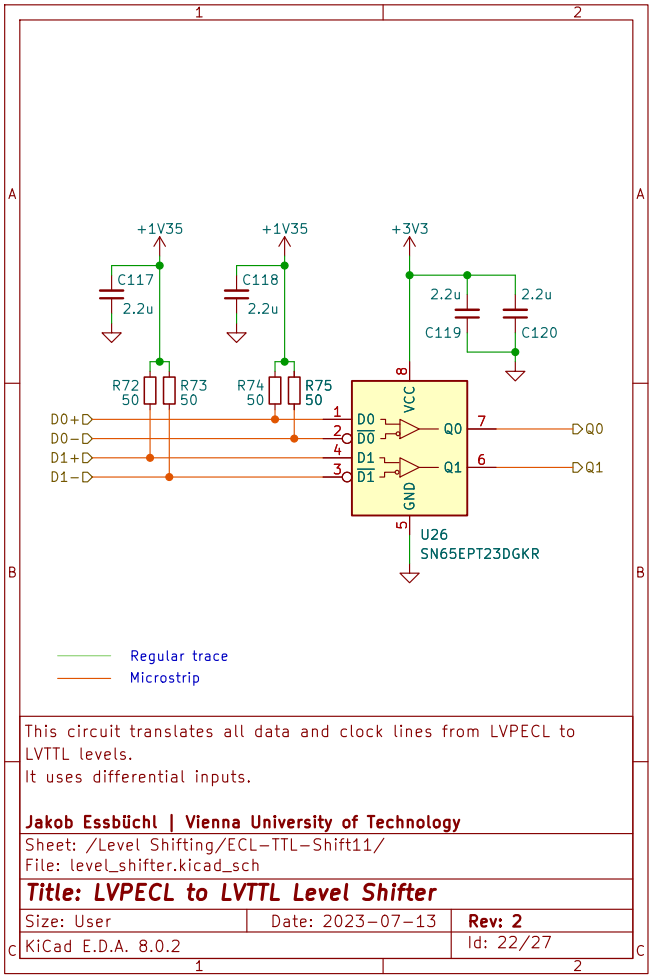
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift10/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 21/27



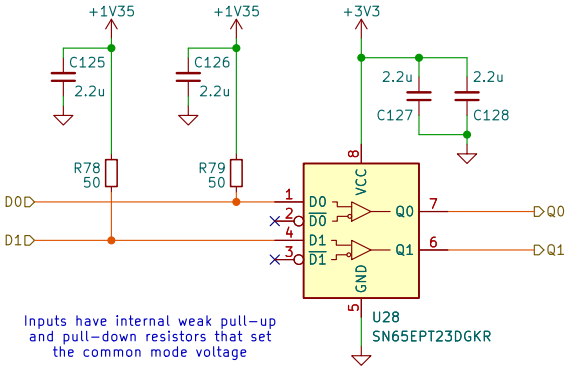
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses differential inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift11/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 22/27



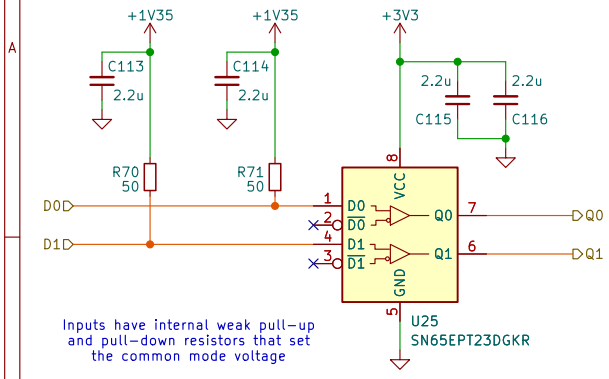
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift12/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 23/27



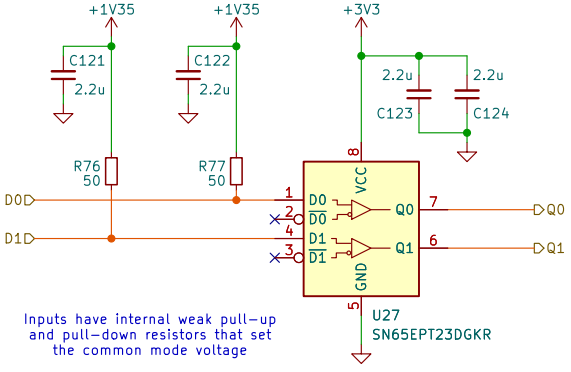
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift13/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 24/27



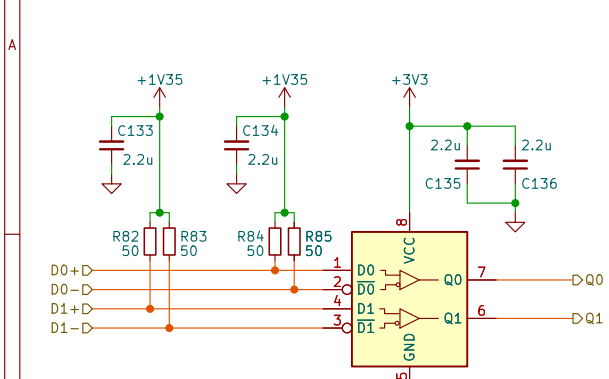
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift14/
File: level_shifter_single_ended.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 25/27



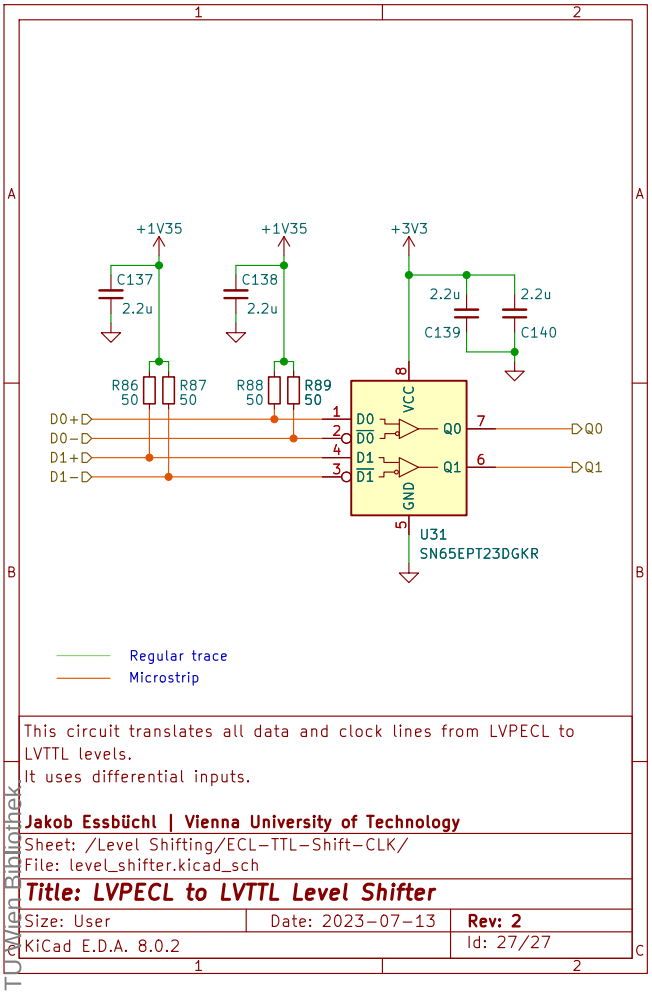
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses differential inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift15/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 26/27



This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses differential inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting/ECL-TTL-Shift-CLK/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2023-07-13	**Rev: 2**
KiCad E.D.A. 8.0.2		Id: 27/27

D.4. Shift Register PCB Layout

All layout figures are to scale and viewed from the top.

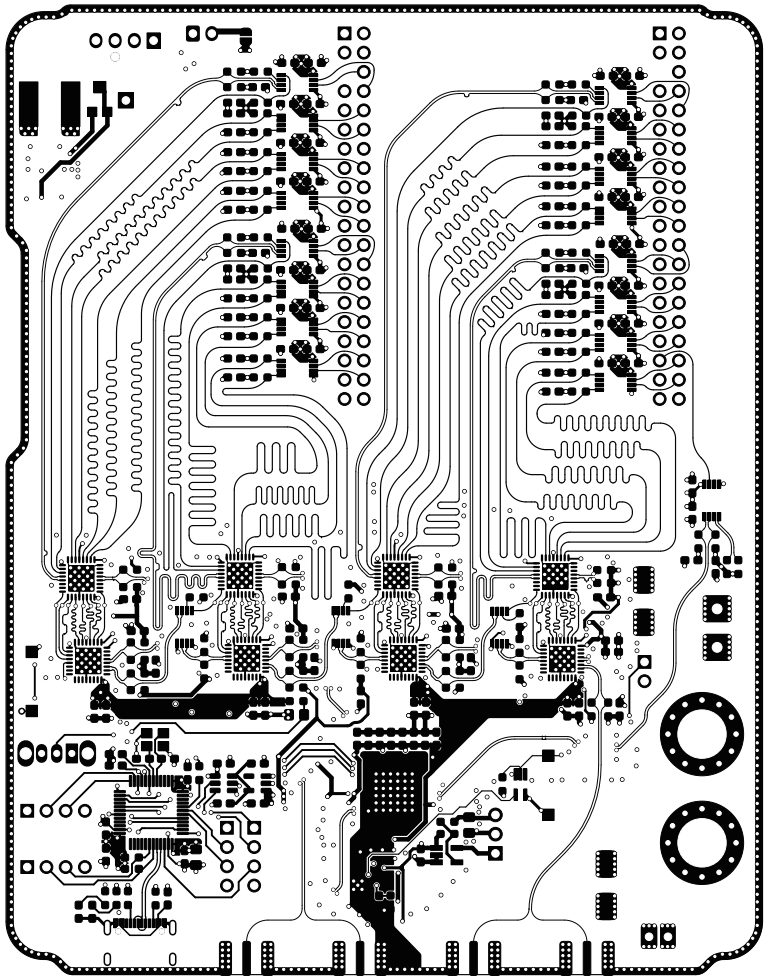


Figure D.7.: Top side copper of the Shift Register PCB. Used for signals, voltage rails, and ground fills.

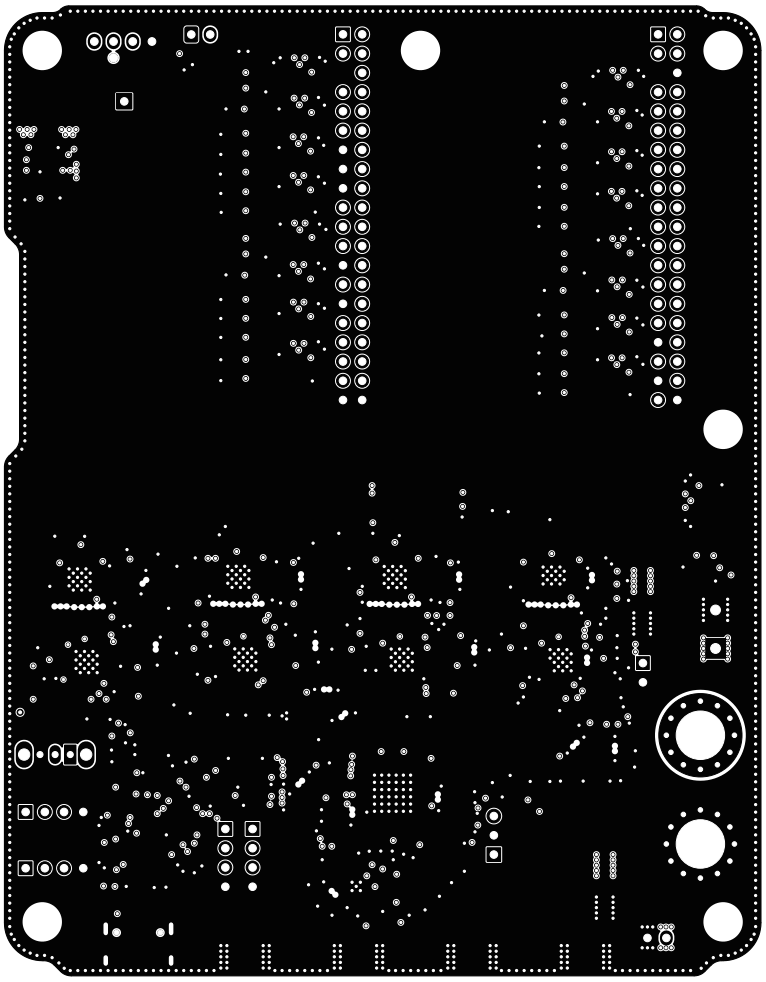


Figure D.8.: Top inner layer (In1) copper of the Shift Register PCB. Used as a ground plane.

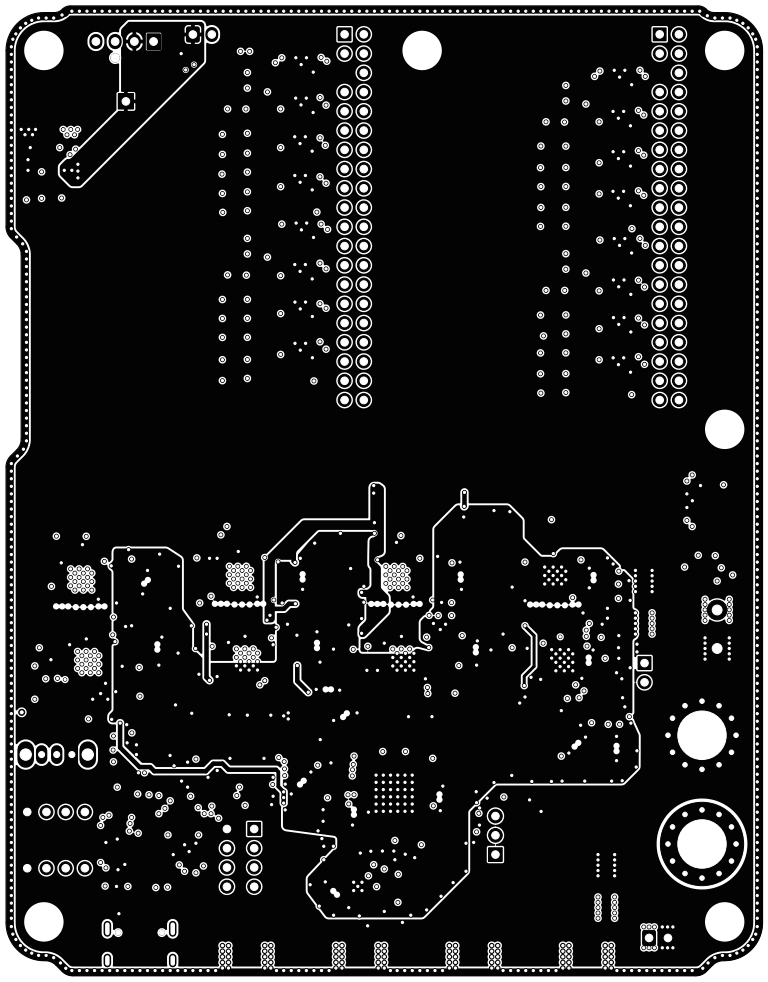


Figure D.9.: Bottom inner layer (In2) copper of the Shift Register PCB. Used for voltage rails as well as a ground island.

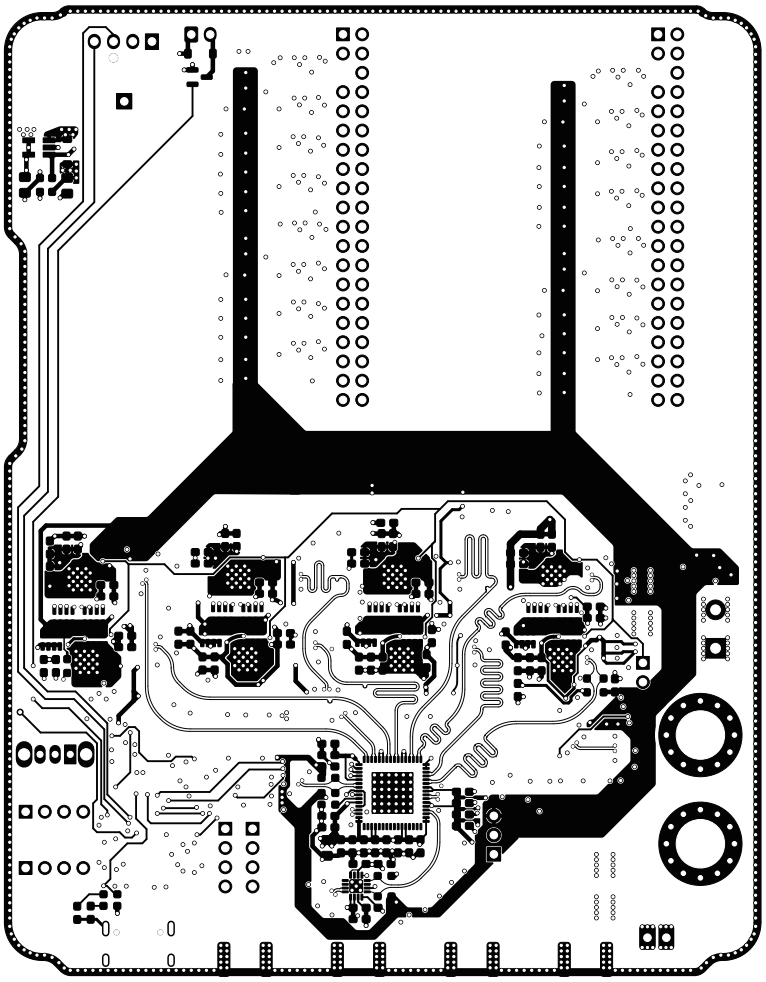


Figure D.10.: Bottom side copper of the Shift Register PCB. Used for signals as well as voltage rails.

D.5. Demux PCB Schematics

The following pages show the schematics as well as the layout of the Demux PCB. Its structure and functionality is described in section 3.3.3 on page 26.

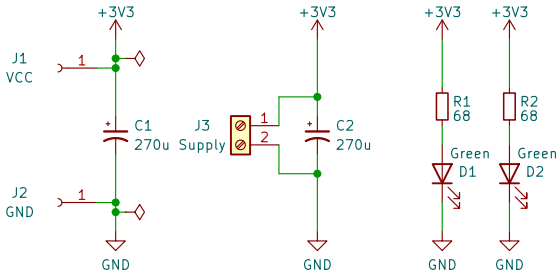
Because the schematics are created with an EDA software (KiCad), they don't feature visible page numbers of the thesis but instead use their own internal numbering system. They are still counted as pages of the thesis however, so pages after the schematics feature the correct page number.

Note that the schematics follow a hierarchical design. Components have a yellow background colour, while hierarchical sheets have a white background colour (see fig. D.11). This is primarily a helpful tool in the used EDA software (KiCad) because it allows functional abstraction as well as reuse of repeating parts of the schematic.

(a) Component (b) Hierarchical sheet

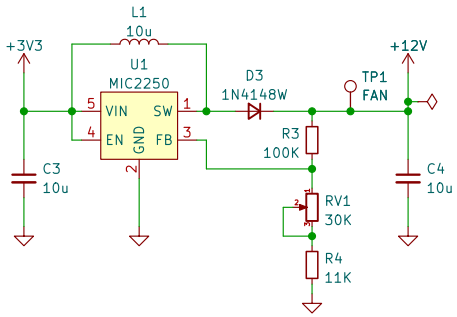
Figure D.11.: Examples for different schematic symbol types.

Power IN

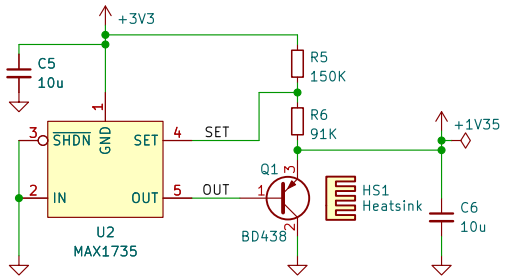


Voltage Regulators

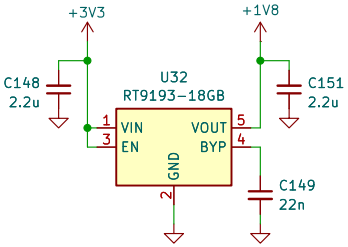
Adjustable 5–12V boost regulator for cooling fan



Regulator for ECL termination voltage VCC–2V
https://www.analog.com/en/design-notes/modified-ldo-regulator-sinks-pecltermination-current.html

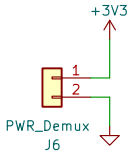


LVPECL common mode voltage for connecting single-ended outputs to differential inputs

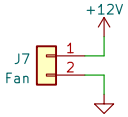


Power OUT

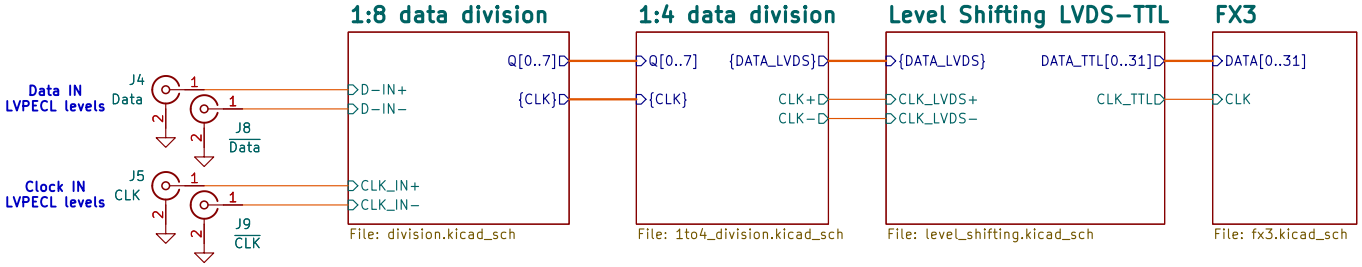
JST-XH connector
to supply power
to comparator PCB



JST-XH connector
for cooling fan



Input from
Comparator PCB



Regular trace
Microstrip

H1 M3 H2 M3 H3 M3 H4 M3

This circuit consists of a 1:8 demultiplexer with 8 1:4 deserializers cascaded behind it.
It converts a differential high-speed serial data stream into 32 parallel data lines.
The associated clock signal is divided by 32. Both are then sent to the FX3 USB interface.

Jakob Essbüchl | Vienna University of Technology

Sheet: /
File: demux-shield.kicad_sch

Title: Demux FX3 Interface

Size: A4	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 1/30

1:8 Demultiplexer

1:8 Clock Fanout

Supervisor for automatic reset after power up

U4 AP1702

SW1 RESET

SW2 RESET

U5 MC100EP445MNG

HSP1 HS2 Heatsink

U6 NB7L1008

HS3 Heatsink

CLK_IN+ CLK_IN−

D−IN+ D−IN−

Q[0..7]

{CLK}

CLK+ CLK−

This circuit takes a high−frequency (<3.3 Gbps) data signal and divides it to 8 parallel lines. The 1/8 clock output of the demux is further routed into a fanout buffer to produce 8 parallel differential clock outputs.
Inputs and outputs are using LVPECL signal levels.

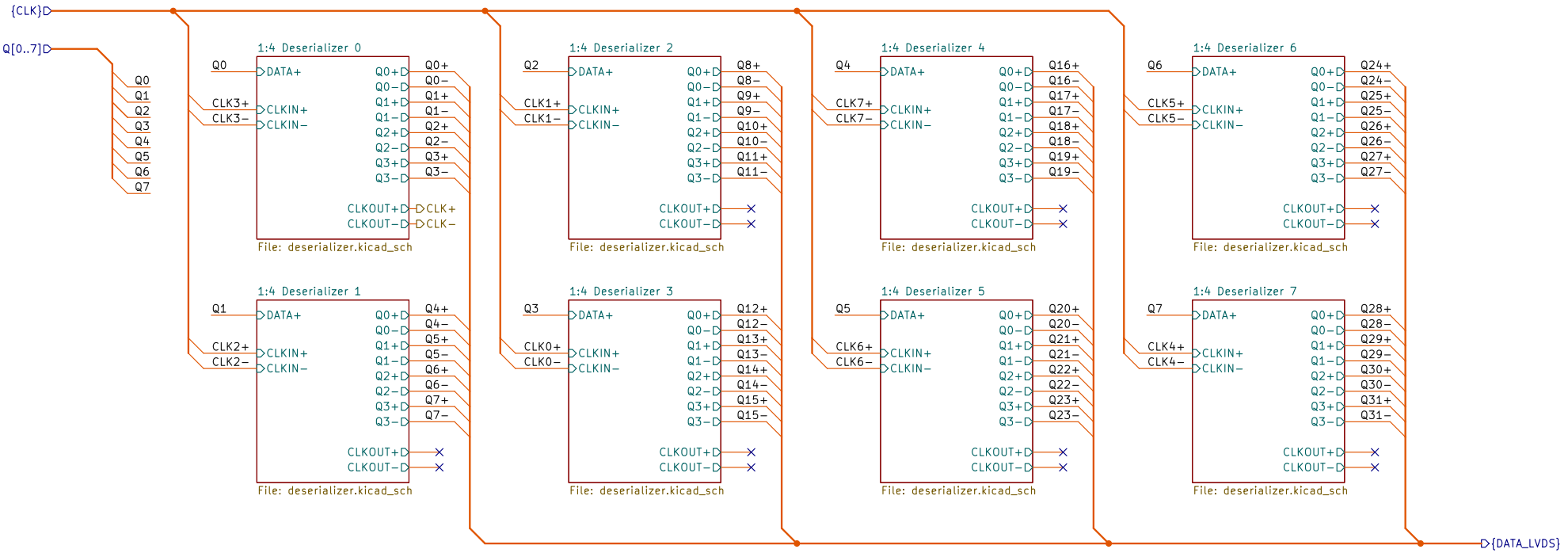
Jakob Essbüchl | Vienna University of Technology

Sheet: /1:8 data division/
File: division.kicad_sch

Title: 1:8 data division

Size: A4	Date: 2024−06−05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 2/30

Regular trace
Microstrip



Regular trace
Microstrip

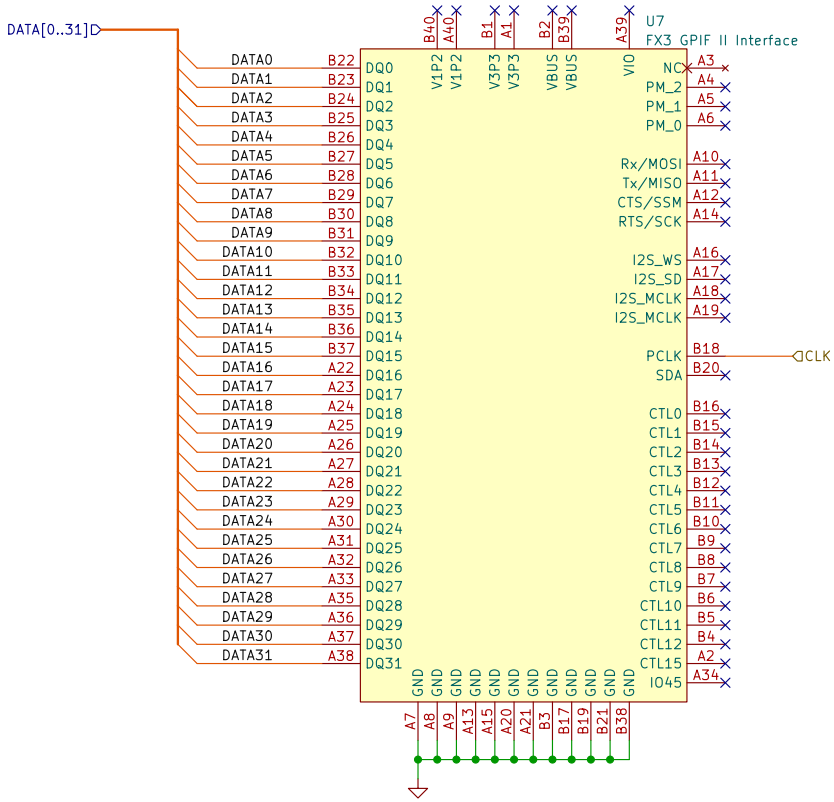
This circuit takes the 8 parallel lines from the Demultiplexer and further divides each into 4 parallel lines, creating 32 parallel data lines in total.
Only one differential clock output is needed for the FX3 further down the signal path, so the clock outputs of the 7 other deserializers are terminated for proper operation but unused.

Jakob Essbüchl | Vienna University of Technology

Sheet: /1:4 data division/
File: 1to4_division.kicad_sch

Title: 1:4 data division

Size: A4	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 3/30



Regular trace
Microstrip

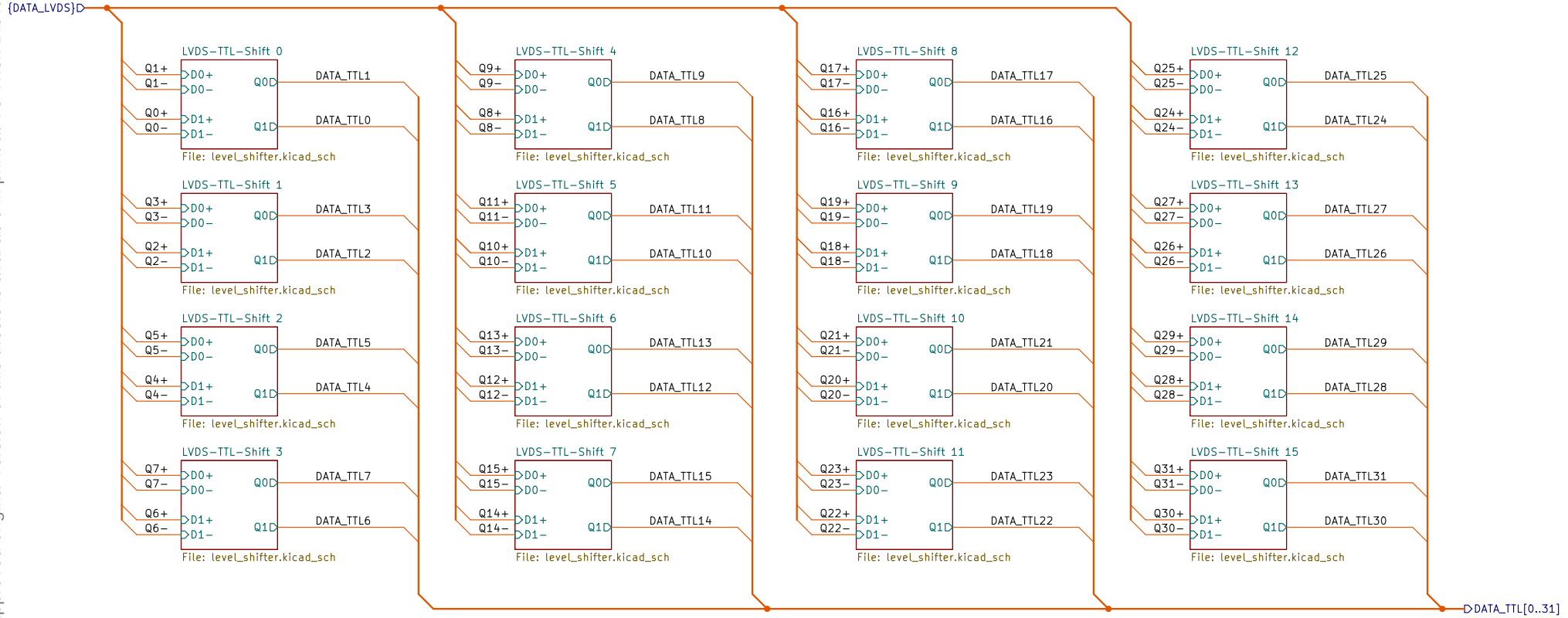
The FX3 facilitates the USB3 connection to the PC.
It connects to the PCB via it's GPIFII interface.

Jakob Essbüchl | Vienna University of Technology

Sheet: /FX3/
File: fx3.kicad_sch

Title: FX3

Size: A4	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 12/30



Regular trace
Microstrip

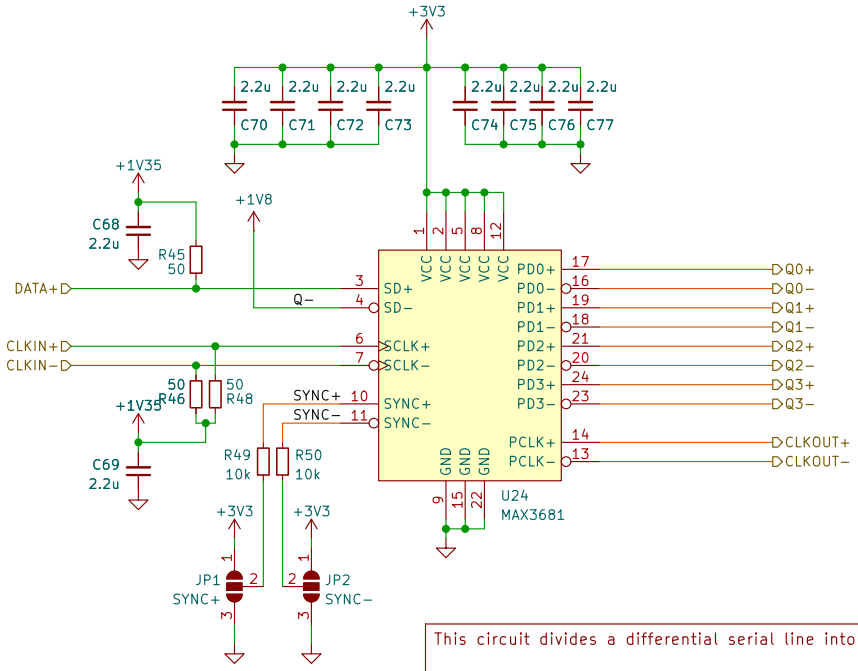
This circuit translates all data and clock lines from LVDS to LVTTL levels.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS-TTL/
File: level_shifting.kicad_sch

Title: Level Shifting LVDS-TTL

Size: A4	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 13/30



This circuit divides a differential serial line into 8 parallel lines.

Inputs are using LVPECL signal levels, outputs are using LVDS signal levels.

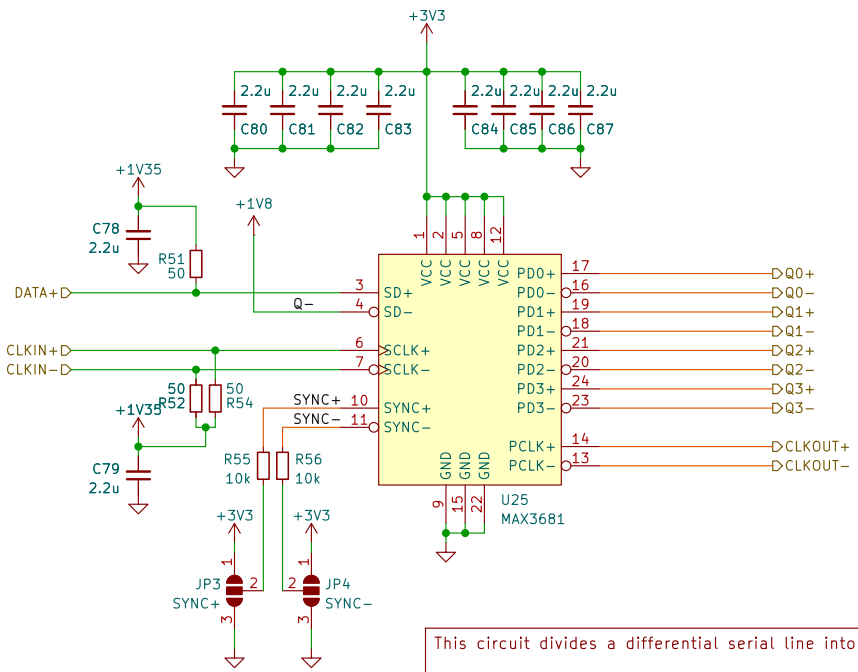
Jakob Essbüchl | Vienna University of Technology

Sheet: /1:4 data division/1:4 Deserializer 0/
File: deserializer.kicad_sch

Title: 1:4 deserializer

Size: A5	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 4/30

Regular trace
Microstrip



This circuit divides a differential serial line into 8 parallel lines.

Inputs are using LVPECL signal levels, outputs are using LVDS signal levels.

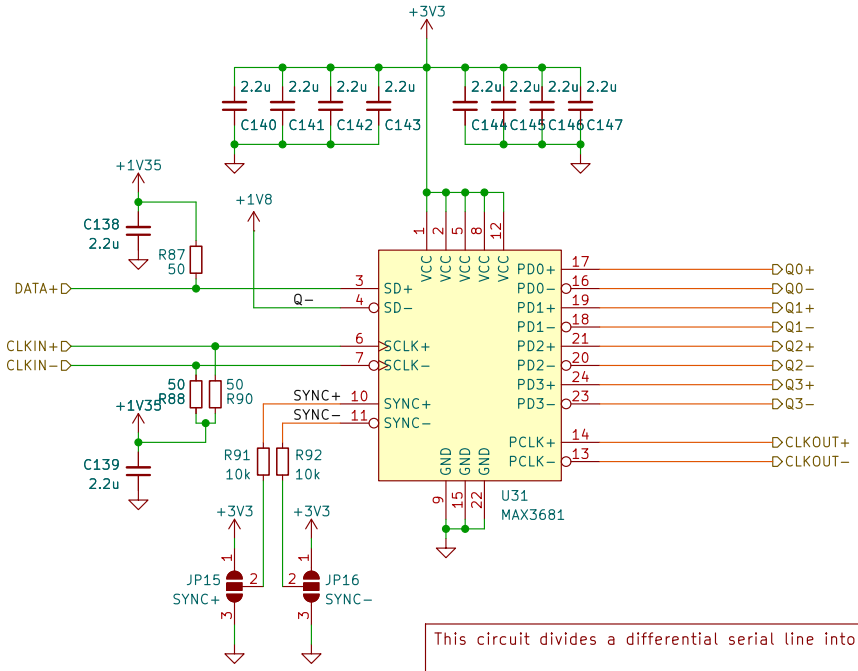
Jakob Essbüchl | Vienna University of Technology

Sheet: /1:4 data division/1:4 Deserializer 1/
File: deserializer.kicad_sch

Title: 1:4 deserializer

Size: A5	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 5/30

Regular trace
Microstrip



This circuit divides a differential serial line into 8 parallel lines.

Inputs are using LVPECL signal levels, outputs are using LVDS signal levels.

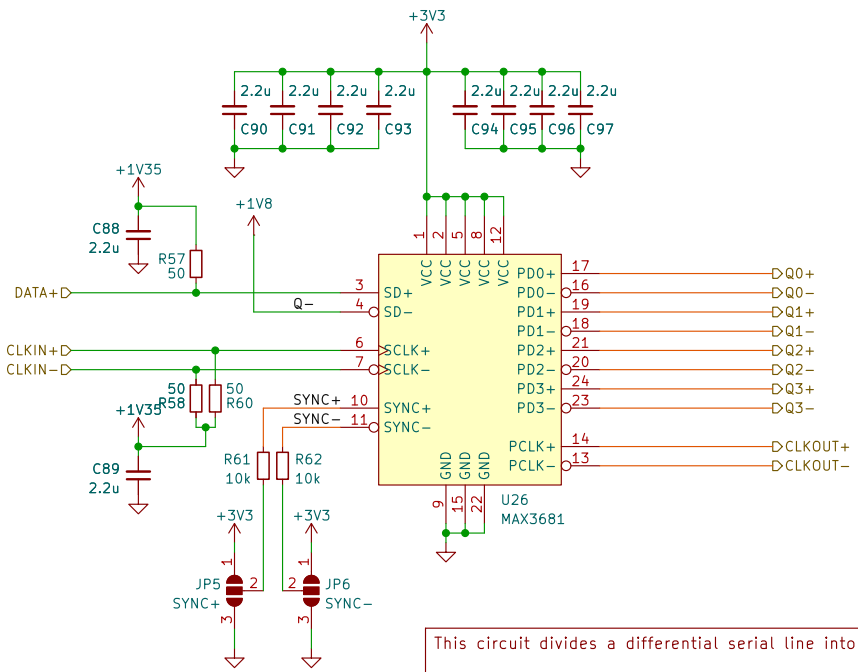
Jakob Essbüchl | Vienna University of Technology

Sheet: /1:4 data division/1:4 Deserializer 2/
File: deserializer.kicad_sch

Title: 1:4 deserializer

Size: A5	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 6/30

Regular trace
Microstrip



This circuit divides a differential serial line into 8 parallel lines.

Inputs are using LVPECL signal levels, outputs are using LVDS signal levels.

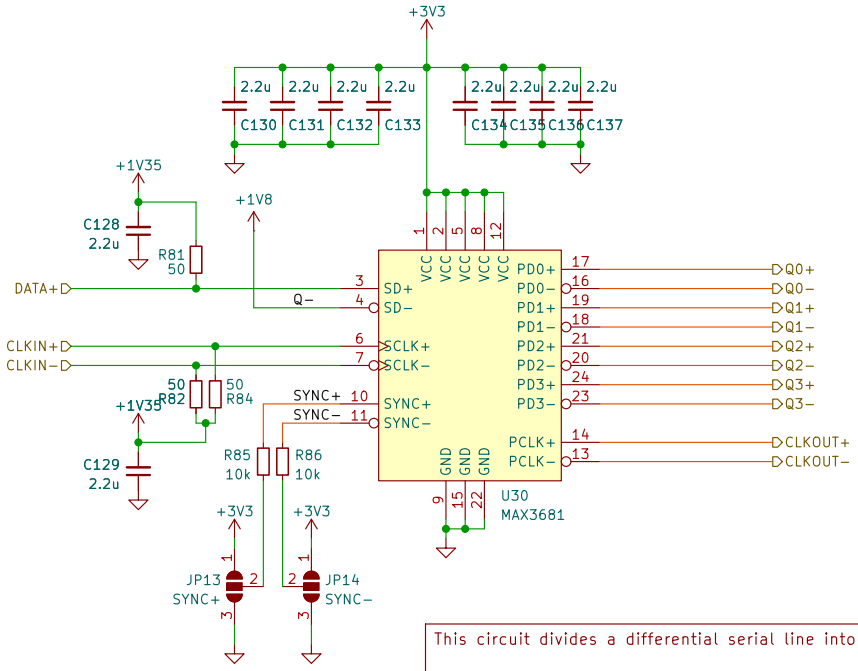
Jakob Essbüchl | Vienna University of Technology

Sheet: /1:4 data division/1:4 Deserializer 3/
File: deserializer.kicad_sch

Title: 1:4 deserializer

Size: A5	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 7/30

Regular trace
Microstrip



This circuit divides a differential serial line into 8 parallel lines.

Inputs are using LVPECL signal levels, outputs are using LVDS signal levels.

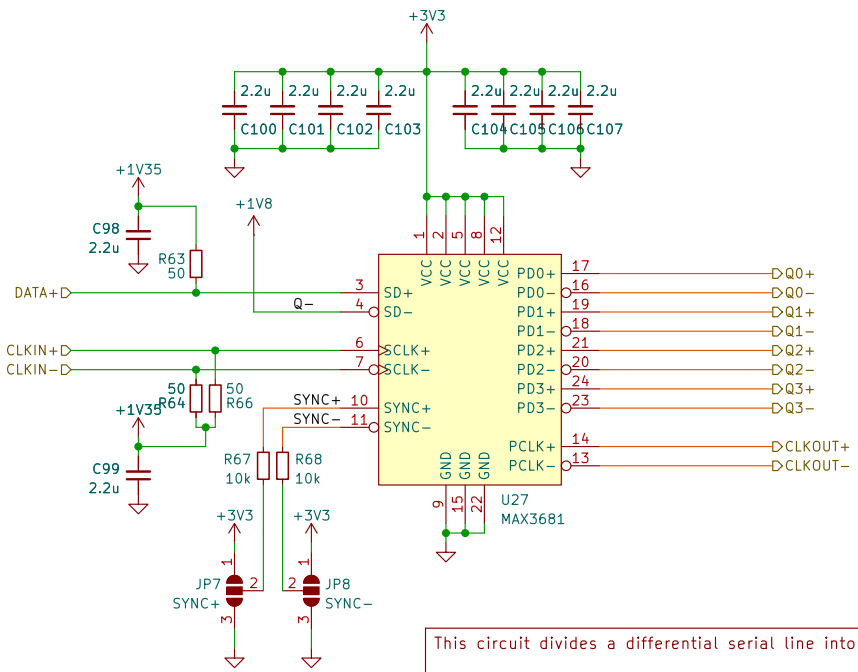
Jakob Essbüchl | Vienna University of Technology

Sheet: /1:4 data division/1:4 Deserializer 4/
File: deserializer.kicad_sch

Title: 1:4 deserializer

Size: A5	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 8/30

Regular trace
Microstrip



This circuit divides a differential serial line into 8 parallel lines.

Inputs are using LVPECL signal levels, outputs are using LVDS signal levels.

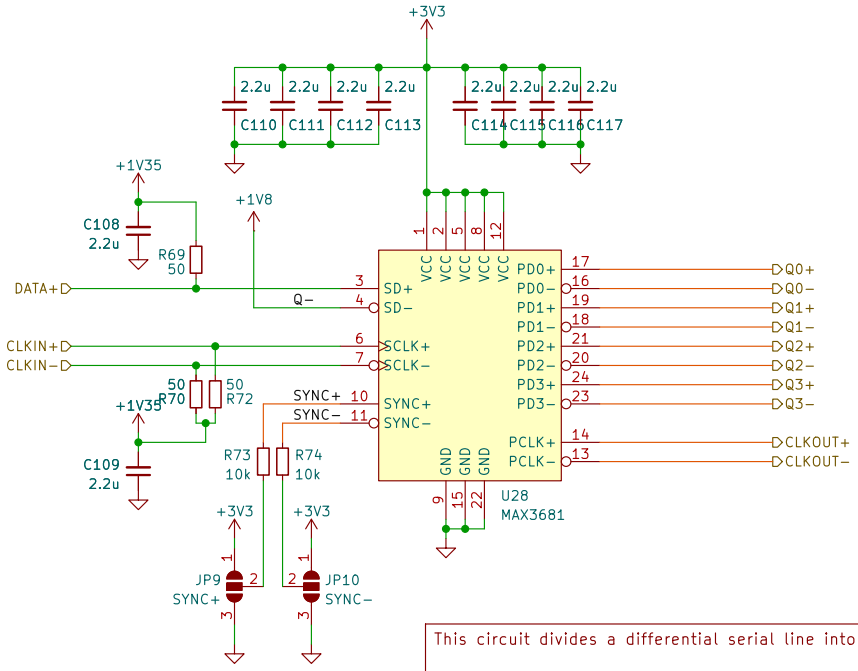
Jakob Essbüchl | Vienna University of Technology

Sheet: /1:4 data division/1:4 Deserializer 5/
File: deserializer.kicad_sch

Title: 1:4 deserializer

Size: A5	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 9/30

Regular trace
Microstrip



This circuit divides a differential serial line into 8 parallel lines.

Inputs are using LVPECL signal levels, outputs are using LVDS signal levels.

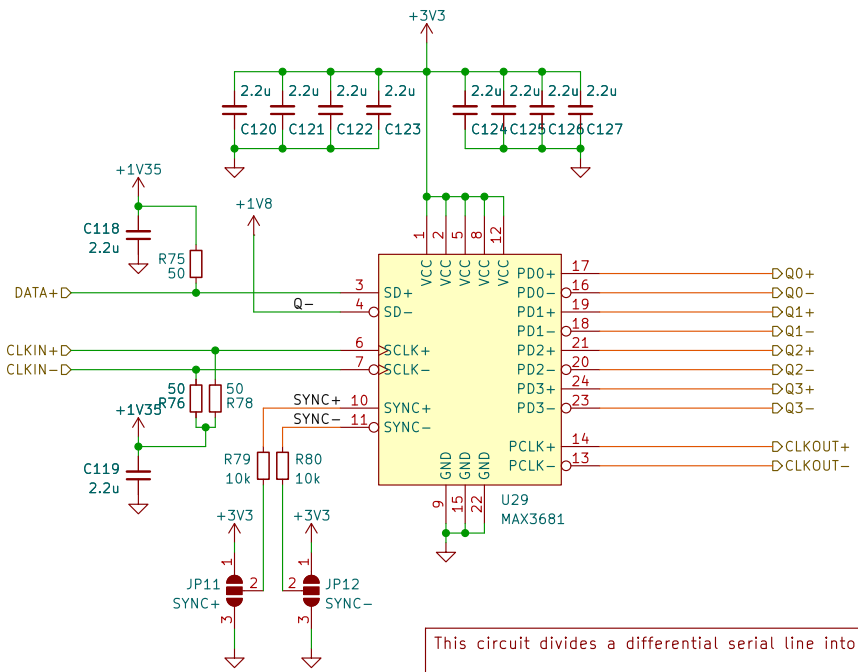
Jakob Essbüchl | Vienna University of Technology

Sheet: /1:4 data division/1:4 Deserializer 6/
File: deserializer.kicad_sch

Title: 1:4 deserializer

Size: A5	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 10/30

Regular trace
Microstrip



This circuit divides a differential serial line into 8 parallel lines.

Inputs are using LVPECL signal levels, outputs are using LVDS signal levels.

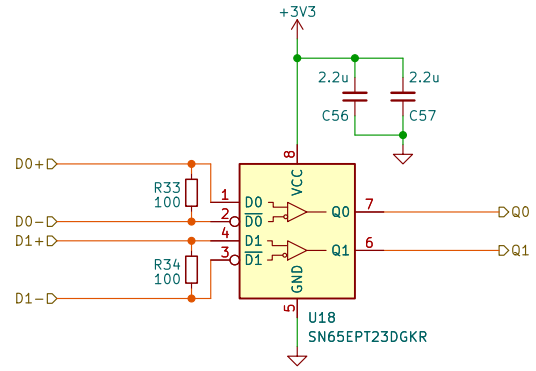
Jakob Essbüchl | Vienna University of Technology

Sheet: /1:4 data division/1:4 Deserializer 7/
File: deserializer.kicad_sch

Title: 1:4 deserializer

Size: A5	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 11/30

Regular trace
Microstrip



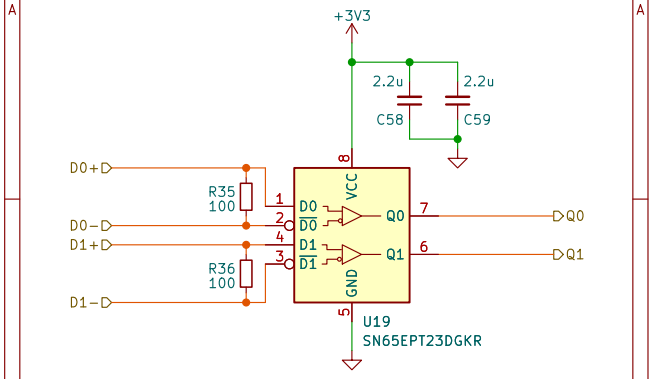
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS−TTL/LVDS−TTL−Shift 0/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024−06−05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 14/30



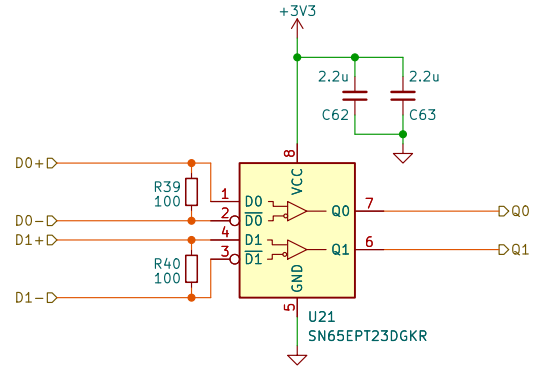
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS−TTL/LVDS−TTL−Shift 1/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024−06−05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 15/30



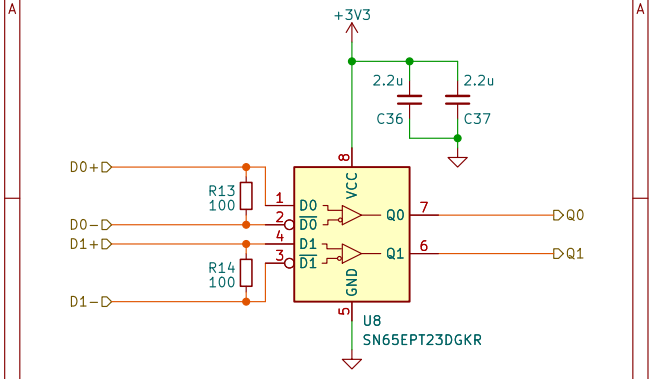
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS−TTL/LVDS−TTL−Shift 2/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024−06−05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 16/30



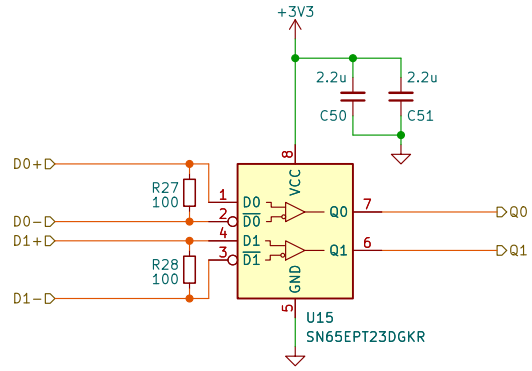
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS−TTL/LVDS−TTL−Shift 3/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024−06−05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 17/30



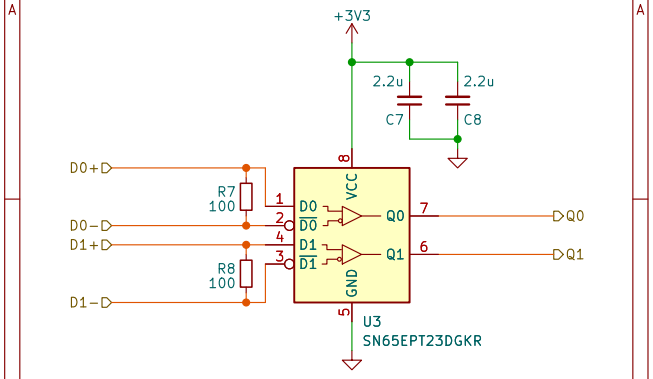
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS–TTL/LVDS–TTL–Shift 4/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024–06–05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 18/30



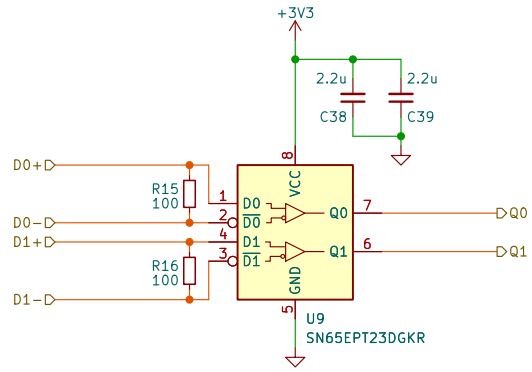
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS–TTL/LVDS–TTL–Shift 5/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024–06–05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 19/30



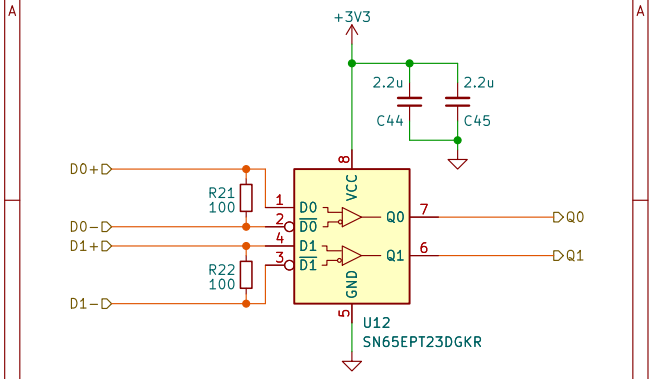
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS–TTL/LVDS–TTL–Shift 6/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024–06–05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 20/30



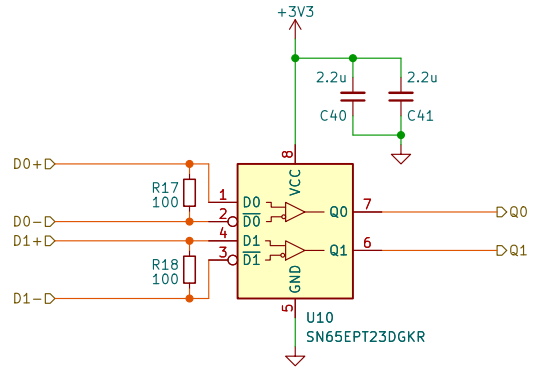
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS–TTL/LVDS–TTL–Shift 7/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024–06–05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 21/30



Regular trace
Microstrip

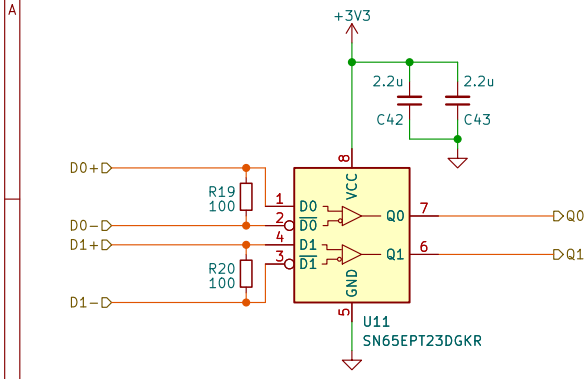
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS–TTL/LVDS–TTL–Shift 8/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024–06–05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 22/30



Regular trace
Microstrip

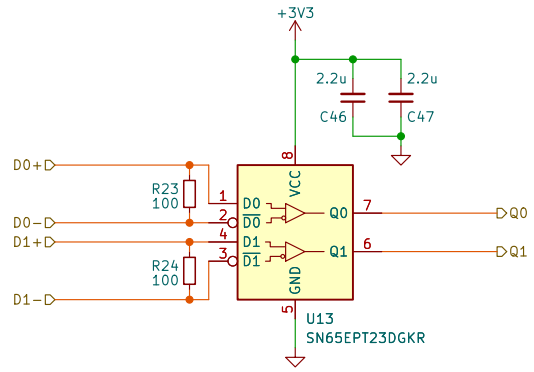
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS–TTL/LVDS–TTL–Shift 9/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024–06–05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 23/30



Regular trace
Microstrip

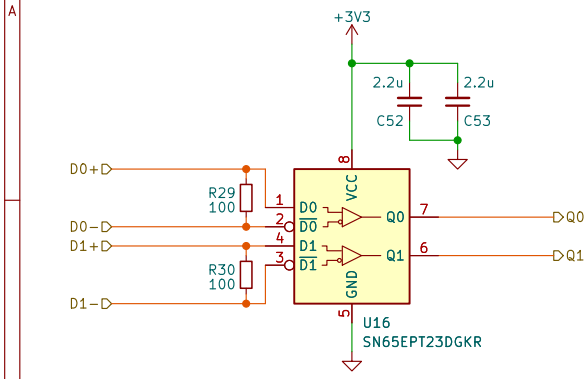
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS–TTL/LVDS–TTL–Shift 10/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024–06–05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 24/30



Regular trace
Microstrip

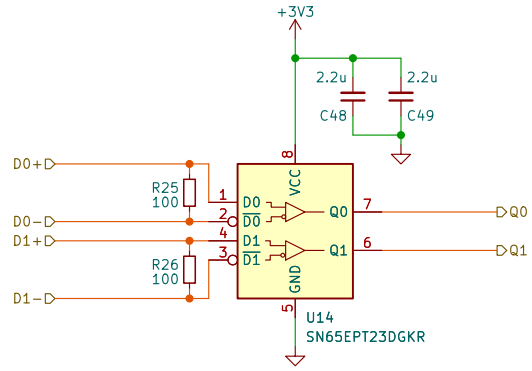
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS–TTL/LVDS–TTL–Shift 11/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024–06–05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 25/30



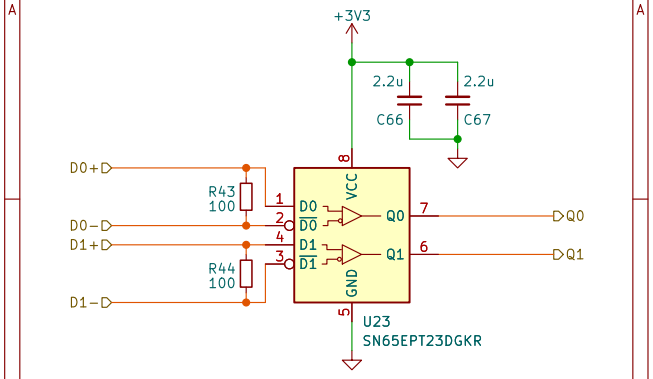
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS-TTL/LVDS-TTL-Shift 12/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 26/30



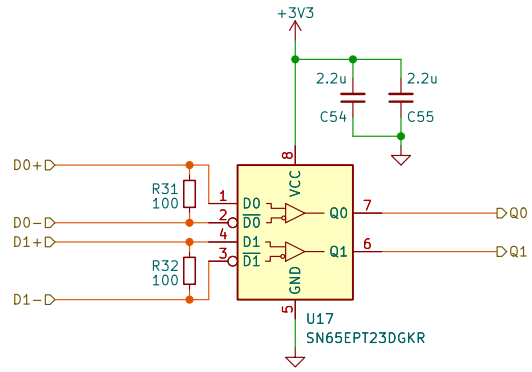
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS-TTL/LVDS-TTL-Shift 13/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 27/30



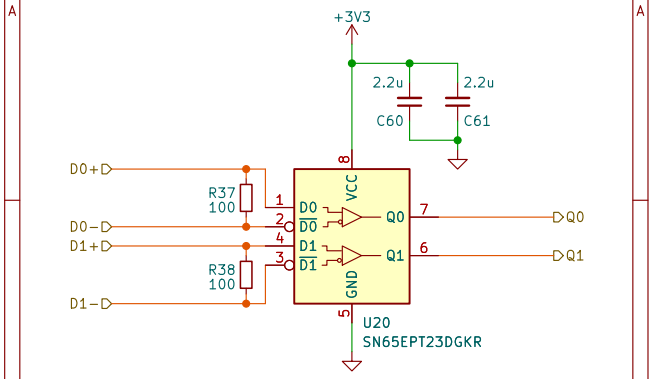
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS-TTL/LVDS-TTL-Shift 14/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 28/30



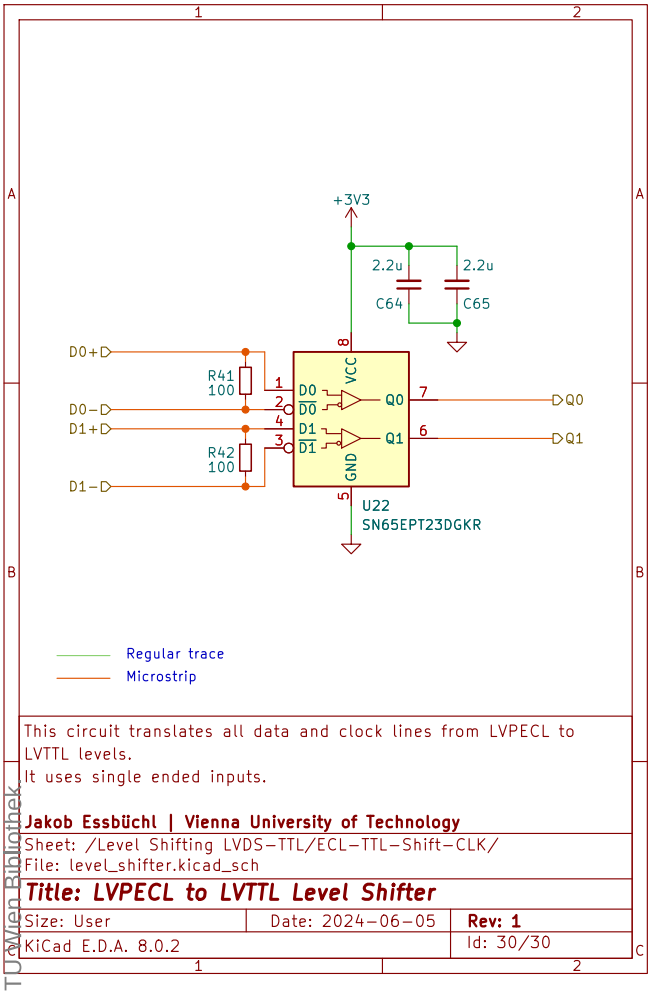
This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS-TTL/LVDS-TTL-Shift 15/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 29/30



This circuit translates all data and clock lines from LVPECL to LVTTL levels.
It uses single ended inputs.

Jakob Essbüchl | Vienna University of Technology

Sheet: /Level Shifting LVDS-TTL/ECL-TTL-Shift-CLK/
File: level_shifter.kicad_sch

Title: LVPECL to LVTTL Level Shifter

Size: User	Date: 2024-06-05	**Rev: 1**
KiCad E.D.A. 8.0.2		Id: 30/30

D.6. Demux PCB Layout

All layout figures are to scale and viewed from the top.

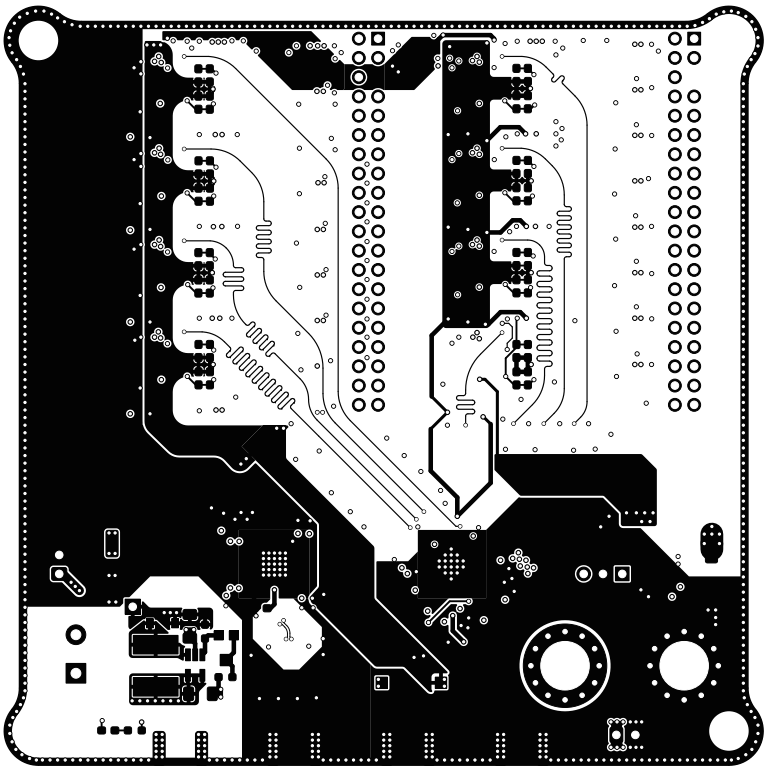


Figure D.12.: Top side copper of the Demux PCB. Used for signals, voltage rails, and ground fills.

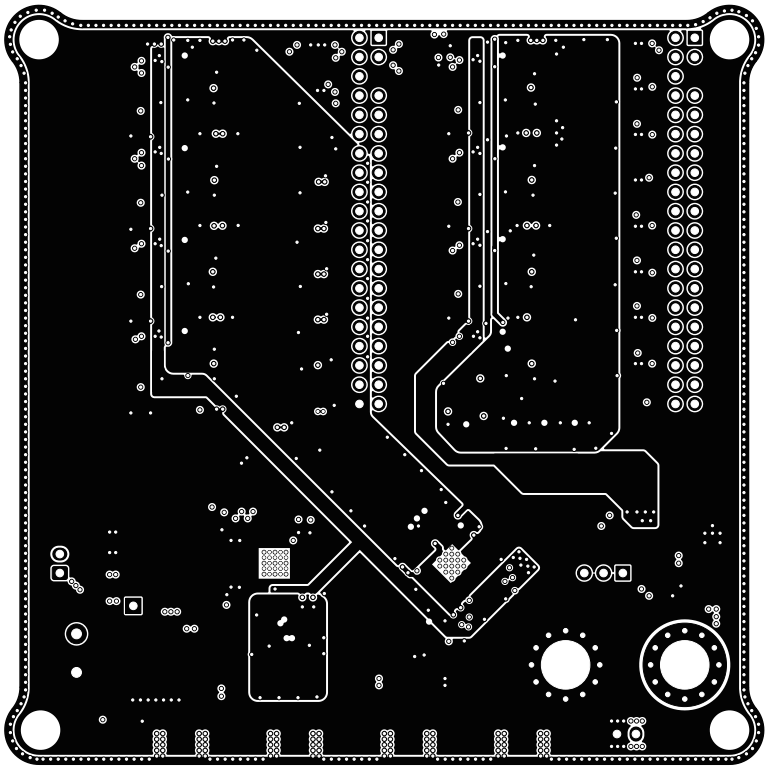


Figure D.13.: Top inner layer (In1) copper of the Demux PCB. Used for voltage rails as well as ground islands.

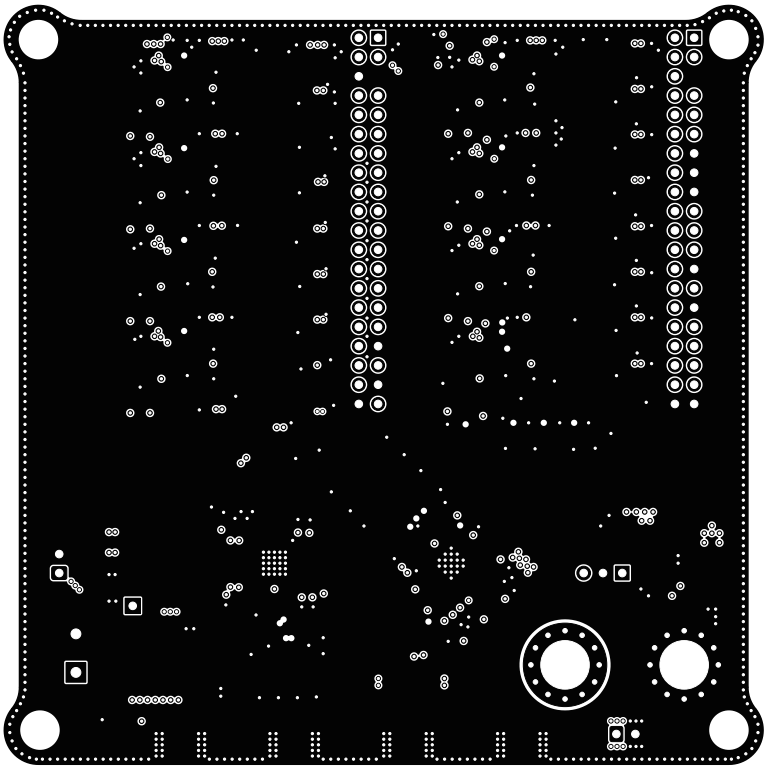


Figure D.14.: Bottom inner layer (In2) copper of the Demux PCB. Used as a ground plane.

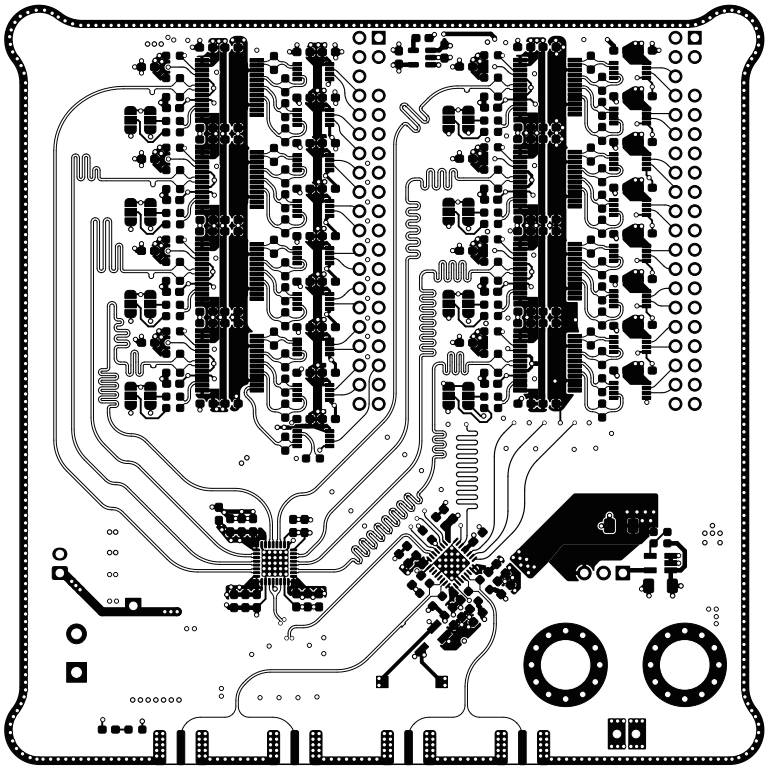


Figure D.15.: Bottom side copper of the Demux PCB. Used for signals as well as voltage rails.

Bibliography

[1] Infineon Technologies AG. 'CYUSB3KIT-003 EZ-USB™ FX3 SuperSpeed explorer kit'. (27th Oct. 2022), [Online]. Available: https://www.infineon.com/cms/en/product/evaluation-boards/cyusb3kit-003/ (visited on 13/06/2024).

[2] H. Zimmermann, *Silicon Optoelectronic Integrated Circuits* (Springer Series in Advanced Microelectronics 13), 2nd ed. 2018. Cham: Springer International Publishing : Imprint: Springer, 2018, 1 p., ISBN: 978-3-030-05822-7. DOI: 10.1007/978-3-030-05822-7.

[3] S. Cova, M. Ghioni, A. Lacaita, C. Samori and F. Zappa, 'Avalanche photodiodes and quenching circuits for single-photon detection', *Applied Optics*, vol. 35, no. 12, p. 1956, 20th Apr. 1996, ISSN: 0003-6935, 1539-4522. DOI: 10.1364/AO.35.001956.

[4] H. Dautet, P. Deschamps, B. Dion *et al.*, 'Photon counting techniques with silicon avalanche photodiodes', *Applied Optics*, vol. 32, no. 21, p. 3894, 20th Jul. 1993, ISSN: 0003-6935, 1539-4522. DOI: 10.1364/AO.32.003894.

[5] H. Finkelstein, M. Hsu and S. Esener, 'STI-Bounded Single-Photon Avalanche Diode in a Deep-Submicrometer CMOS Technology', *IEEE Electron Device Letters*, vol. 27, no. 11, pp. 887–889, Nov. 2006, ISSN: 0741-3106, 1558-0563. DOI: 10.1109/LED.2006.883560.

[6] B. Behroozpour, P. A. M. Sandborn, M. C. Wu and B. E. Boser, 'Lidar System Architectures and Circuits', *IEEE Communications Magazine*, vol. 55, no. 10, pp. 135–142, Oct. 2017, ISSN: 1558-1896. DOI: 10.1109/MCOM.2017.1700030.

[7] A. Restelli, J. C. Bienfang and A. L. Migdall, 'Time-domain measurements of afterpulsing in InGaAs/InP SPAD gated with sub-nanosecond pulses', *Journal of Modern Optics*, vol. 59, no. 17, pp. 1465–1471, 10th Oct. 2012, ISSN: 0950-0340, 1362-3044. DOI: 10.1080/09500340.2012.687463.

[8] A. W. Ziarkash, S. K. Joshi, M. Stipčević and R. Ursin, 'Comparative study of afterpulsing behavior and models in single photon counting avalanche photo diode detectors', *Scientific Reports*, vol. 8, no. 1, p. 5076, 22nd Mar. 2018, ISSN: 2045-2322. DOI: 10.1038/s41598-018-23398-z.

[9] Electronic Industries Association, 'High Speed Transceiver Logic (HSTL) A 1.5 V Output Buffer Supply Voltage Based Interface Standard for Digital Integrated Circuits', EIA/JESD8-6, Aug. 1995, p. 13.

[10] Renesas Electronics Corporation, 'I/O Interface Standards', Renesas Electronics Corporation, Tokyo, Japan, AN-230, Jun. 2004. [Online]. Available: https://www.renesas.com/us/en/document/apn/230-io-interface-standards (visited on 25/08/2024).

[11] Zhaoming Wu, C. Zhang, F. Li and Z. Wang, 'High speed serial interface transceiver controller based on JESD204B', in *2016 14th IEEE International New Circuits and Systems Conference (NEWCAS)*, Vancouver, BC, Canada: IEEE, Jun. 2016, pp. 1–4, ISBN: 978-1-4673-8900-6. DOI: 10.1109/NEWCAS.2016.7604778.

[12] Texas Instruments, 'LVDS Owner's Manual', Texas Instruments Incorporated, Dallas, TX, USA, Fourth Edition, 2008, p. 109. [Online]. Available: https://www.ti.com/lit/ug/snla187/snla187.pdf (visited on 27/07/2024).

[13] M. Alioto and G. Palumbo, *Model and Design of Bipolar and MOS Current-Mode Logic: CML, ECL and SCL Digital Circuits*. Boston, MA: Springer US, 2005, ISBN: 978-1-4020-2878-6 978-1-4020-2888-5. DOI: 10.1007/1-4020-2888-1.

[14] P. Heydari, 'Design and analysis of low-voltage current-mode logic buffers', in *Fourth International Symposium on Quality Electronic Design, 2003. Proceedings.*, Mar. 2003, pp. 293–298. DOI: 10.1109/ISQED.2003.1194748.

[15] H. S. Yourke, 'Transistor switching circuits', U.S. Patent 2964652A, 13th Dec. 1960. [Online]. Available: https://patents.google.com/patent/US2964652A/en (visited on 25/08/2024).

[16] JEDEC Solid State Technology Association, 'Serial Interface for Data Converters', Arlington, VA, USA, JESD204D, Dec. 2023, p. 136.

[17] P. Horowitz, *The Art of Electronics*, Third edition. New York, NY, USA: Cambridge University Press, 2015, 1192 pp., ISBN: 978-0-521-80926-9.

[18] T. Dreier, D. Krapohl, D. Maneuski *et al.*, 'A USB 3.0 readout system for Timepix3 detectors with on-board processing capabilities', *Journal of Instrumentation*, vol. 13, no. 11, pp. C11017–C11017, 23rd Nov. 2018, ISSN: 1748-0221. DOI: 10.1088/1748-0221/13/11/C11017.

[19] B. Janssen, M. Hubner and T. Jaeschke, 'An AXI compatible cypress EZ-USB FX3 interface for USB-3.0 SuperSpeed', in *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*, Cancun: IEEE, Dec. 2014, pp. 1–4, ISBN: 978-1-4799-5944-0. DOI: 10.1109/ReConFig.2014.7032498.

[20] Y. Li, K. Song, K. Zhong, J. Liang and H. Liu, 'Design of High-Speed Data Transmission System Based on USB 3.0', *IEEE Transactions on Nuclear Science*, vol. 70, no. 6, pp. 1090–1095, Jun. 2023, ISSN: 1558-1578. DOI: 10.1109/TNS.2023.3240542.

[21] D. Park, J. Yoon and J. Kim, 'A low-power SerDes for high-speed on-chip networks', in *2017 International SoC Design Conference (ISOCC)*, Nov. 2017, pp. 252–253. DOI: 10.1109/ISOCC.2017.8368879.

[22] F. Tobajas, R. Esper-Chain, R. Regidor, O. Santana and R. Sarmiento, 'A Low Power 2.5 Gbps 1:32 Deserializer in SiGe BiCMOS Technology', in *2006 IEEE Design and Diagnostics of Electronic Circuits and Systems*, Apr. 2006, pp. 19–24. DOI: 10.1109/DDECS.2006.1649564.

[23] J. Kim and H. Shin, 'A low-power 3.52 gbps serdes with a mdll frequency multiplier for high-speed on-chip networks', *Journal of Semiconductor Technology and Science*, vol. 18, no. 6, pp. 658–666, 2018. DOI: 10.5573/JSTS.2018.18.6.658.

[24] Infineon Technologies AG, 'EZ-USB FX3 SuperSpeed USB Controller', CYUSB301X, CYUSB201X Datasheet Rev. *Z, 29th Sep. 2022. [Online]. Available: https://www.infineon.com/dgdl/Infineon-CYUSB301X_CYUSB201X_EZ-USB_FX3_SUPERSPEED_USB_CONTROLLER-DataSheet-v21_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0eca1e7442aa (visited on 13/06/1924).

[25] J. Hyde, *SuperSpeed Device Design by Example: USB Design by Example*, 2. ed. s.l.: CreateSpace, 2015, 246 pp., ISBN: 978-1-5005-8805-2.

[26] Infineon Technologies AG, 'How to implement an image sensor interface using EZ-USB™ FX3 in a USB Video Class (UVC) framework', Infineon Technologies AG, Munich, Germany, AN75779 - Rev. *L, 27th Oct. 2021. [Online]. Available: https://www.infineon.com/dgdl/Infineon-AN75779_How_to_Implement_an_Image_Sensor_Interface_with_EZ-USB_FX3_in_a_USB_Video_Class_(UVC)_Framework-ApplicationNotes-v13_00-EN.pdf?fileId=8ac78c8c7cdc391c017d073ad2b85f0d (visited on 06/04/2024).

[27] Infineon Technologies. 'DMA buffer count and size allocation in EZ-USB™ FX3/CX3'. (14th May 2024), [Online]. Available: https://community.infineon.com/t5/Knowledge-Base-Articles/DMA-buffer-count-and-size-allocation-in-EZ-USB-FX3-CX3/ta-p/248983 (visited on 22/06/2024).

[28] Manaskant Desai and Karthik Sivaramakrishnan, 'Optimizing USB 3.0 Throughput With EZ-USB® FX3™', Infineon Technologies AG, Munich, Germany, AN86947 - Rev *D, 23rd Aug. 2017. [Online]. Available: https://www.infineon.com/dgdl/Infineon-AN86947_Optimizing_USB_3.0_Throughput_with_EZ-USB_FX3-ApplicationNotes-v05_00-EN.pdf?fileId=8ac78c8c7cdc391c017d073e3a2e6243 (visited on 22/06/2024).

[29] Cypress Semiconductor, 'EZ-USB® FX3™ Technical Reference Manual', Rev. *F, 9th May 2019. [Online]. Available: https://www.infineon.com/dgdl/Infineon-EZ-USB_FX3_Technical_Reference_Manual-AdditionalTechnicalInformation-v07_00-EN.pdf?fileId=8ac78c8c7d0d8da4017d0f90b94c7db6 (visited on 07/07/2024).

[30] USB 3.0 Promoter Group (Apple Inc., Hewlett-Packard Inc., Intel Corporation, Microsoft Corporation, Renesas Corporation, STMicroelectronics, and Texas Instruments), 'Universal Serial Bus 3.2 Specification', Revision 1.1, Jun. 2022. [Online]. Available: https://www.usb.org/document-library/usb-32-revision-11-june-2022 (visited on 03/07/2024).

[31] P. Shockman, 'Termination of ECL Devices with EF (Emitter Follower) Output Structure', Semiconductor Components Industries, LLC, Phoenix, AZ, USA, AND8020/D - Rev. 6, Apr. 2007. [Online]. Available: https://www.onsemi.com/pub/Collateral/AND8020-D.PDF (visited on 22/05/2023).

[32] Maxim Integrated Products, 'Modified LDO Regulator Sinks PECL-Termination Current', AN 3627, 25th Sep. 2005. [Online]. Available: https://www.analog.com/media/en/technical-documentation/design-notes/modified-ldo-regulator-sinks-peclterminationcurrent.pdf (visited on 05/07/2024).

[33] H. W. Ott, *Electromagnetic Compatibility Engineering*. Hoboken, N.J, USA: John Wiley & Sons, 2009, 843 pp., ISBN: 978-0-470-18930-6.

[34] D. C. Smith. 'Routing Signals Between PWB Layers - Part 2, An Emissions Example', High Frequency Measurements. (May 2006), [Online]. Available: https://emcesd.com/tt2006/tt050106.htm (visited on 06/07/2024).

[35] Renesas Electronics Corporation, 'JESD204B Compliant Fanout Buffer and Divider', Renesas Electronics Corporation, Tokyo, Japan, 8V79S680 Datasheet, 11th Jan. 2019. [Online]. Available: https://www.renesas.com/us/en/document/dst/8v79s680-datasheet (visited on 05/07/2024).

[36] C. Wolfe, 'Clocking Design Guidelines: Unused Pins', Texas Instruments Incorporated, Dallas, TX, USA, SNVA745, Nov. 2015. [Online]. Available: https://www.ti.com/lit/snva745 (visited on 24/07/2024).

[37] ON Semiconductor, '3.3 V/5 V ECL 8-Bit Serial/Parallel Converter', Semiconductor Components Industries, LLC, Phoenix, AZ, USA, MC100EP455 Datasheet Rev. 17, Apr. 2021. [Online]. Available: https://www.onsemi.com/pdf/datasheet/mc100ep445-d.pdf (visited on 21/07/2024).

[38] B. Goll and H. Zimmermann, *Comparators in Nanometer CMOS Technology* (Springer Series in Advanced Microelectronics). Berlin, Heidelberg, Germany: Springer, 2015, vol. 50, 250 pp., ISBN: 978-3-662-44482-5. DOI: 10.1007/978-3-662-44482-5.

[39] Keysight Technologies. 'Bit Error Rate vs Bit Error Ratio – What is the Difference?', [Online]. Available: https://docs.keysight.com/kkbopen/bit-error-rate-vs-bit-error-ratio-what-is-the-difference-620138607.html (visited on 27/03/2023).

[40] Phabrix. 'Using Pseudo-Random Binary Sequences to Stress Test Serial Digital Interfaces', [Online]. Available: https://phabrix.com/ftp/App_Notes/Stress_Whitepaper.pdf (visited on 27/07/2024).

[41] International Telecommunication Union, *Digital test patterns for performance measurements on digital transmission equipment*, Geneva, Switzerland, Oct. 1992. [Online]. Available: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-O.150-199210-S!!PDF-E&type=items (visited on 27/07/2024).

[42] International Telecommunication Union, *Error performance measuring equipment operating at the primary rate and above*, Geneva, Switzerland, Oct. 1992. [Online]. Available: https://www.itu.int/rec/dologin_pub.asp?lang=f&id=T-REC-O.151-199210-I!!PDF-E&type=items (visited on 27/07/2024).

[43] C. Sterzik, 'Power consumption of LVPECL and LVDS', Texas Instruments Incorporated, Dallas, TX, USA, SLYT127, 1Q 2002. [Online]. Available: https://www.ti.com/lit/an/slyt127/slyt127.pdf (visited on 23/06/2024).

[44] H. W. Johnson and M. Graham, *High Speed Digital Design: A Handbook of Black Magic*, 30. pr. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1993, 447 pp., ISBN: 978-0-13-395724-2.

[45] Analog Devices Inc., 'LVDS/Anything-to-LVPECL/LVDS Dual Translator', MAX9376 Datasheet Rev. 1, Oct. 2009. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX9376.pdf (visited on 06/04/2024).

[46] Analog Devices Inc. 'MAX9376 Product Info', [Online]. Available: https://www.analog.com/en/products/max9376.html (visited on 06/04/2024).

[47] Mark Chang, 'Introduction to IBIS Modeling of Fiber Optic Transceivers', Agilent Technologies, Fiber Optics Communication Division, San Jose, CA, USA, 1st Sep. 2008. [Online]. Available: https://web.archive.org/web/20110707094747/http://cp.literature.agilent.com/litweb/pdf/5990-3107EN.pdf (visited on 06/04/2024).

[48] C. K. A. Rangan, K. A. Holla, V. Kulkarni, A. Kumar and A. Patil, 'Data rate based performance analysis and optimization of bulk OUT transactions in USB 3.0 SuperSpeed protocol', in *2017 3rd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT)*, Tumkur: IEEE, Dec. 2017, pp. 114–119, ISBN: 978-1-5386-1144-9. DOI: 10.1109/ICATCCT.2017.8389117.

[49] mgiacomelli. 'Detecting buffer overflows and recovering', infineon Developer Community. (29th Jul. 2021), [Online]. Available: https://community.infineon.com/t5/USB-superspeed-peripherals/Detecting-buffer-overflows-and-recovering/m-p/284198#M27689 (visited on 07/05/2024).

Glossary

APD Avalanche photodiode. Photodetector that is operated above the diode breakdown voltage. The photocurrent of the diode is strongly amplified by an avalanche process. 3, 30, 44, 58

ASIC Application Specific Integrated Circuit. A custom-designed integrated circuit created for a specific task. Very good performance and efficiency compared to more generalized solutions like FPGAs or CPUs, but has a fixed design/application and high development and manufacturing costs. 6, 9

BER Bit error ratio. Important metric of transfer reliability and performance of digital communications systems. Defined as the ratio of the number of bit errors divided by the total number of transferred bits. Not to be confused with the bit error rate (BER), which is the bit error ratio multiplied by the bit rate and gives bit errors per unit of time. 8, 35–37, 39, 40, 44, 46, 48, 50, 51, 53

DMA Direct Memory Access. Allows accessing system memory directly and independently of the main central processing unit (CPU) 9–14, 16, 17, 55

FPGA Field Programmable Gate Array. An integrated circuit that can be (re)programmed after manufacturing and thus is adjustable for a wide range of digital functions. 9, 45, 46, 53

Geiger mode A name for the way the avalanche process leads to output pulses in single-photon avalanche diodes (SPADs). A reference to the way a Geiger-counter produces 'click' events. 3

GPIF II General programmable interface II 9, 11, 13, 14, 57

GUI Graphical user interface. A type of interface that allows users to interact with devices using visual elements like windows, icons and menus. 55

JTAG Joint Test Action Group. A protocol widely used for debugging microcontrollers and similar internal circuits (ICs) 10

LDO Low dropout. An LDO regulator is a voltage regulator with a lowered dropout voltage compared to regular voltage regulator. 19, 30

LFSR Linear feedback shift register. A circuit commonly used to generate pseudo-random binary sequences (PRBS). 36

PRBS Pseudo-random binary sequence. Binary sequences of varying length that appear random but can be generated in a deterministic manner. Commonly used to measure the bit error ratio (BER). 36, 39, 44, 50, 57

RTOS Real-time operating system. A type of operating system that is able to perform actions within predictable – and often strict – time limits. 13, 52

SDK Software development kit. A collection of tools, libraries and documentations to build applications or firmware. 14, 17, 18

SPAD Single-photon avalanche diode. A highly sensitive photodetector that can detect individual photons through a self-sustaining avalanche process. Conceptually an avalanche photodiode (APD) operated in Geiger mode 1, 3, 4, 7, 8, 23, 29, 30, 39, 44, 53, 57, 58, 61